TAKARA

ゲームパソコン

BASIC-I TEXT

はじめに

BASIC I は、本機で初めてコンピュータに触れる方、これからBASIC言語を勉強しようとされている方のために、今までの BASIC 言語の基本となる部分を集約した「入門用BASIC言語」といえるものです。つまり、今までのBASIC言語でむずかしかった複雑な部分を取ってしまったと考えてください。
BASIC I は、今までのBASIC言語の基本を集約したとはいえ、命令文(ステートメント)は今までのBASIC言語と全く同じです。これは、BASIC I でBASIC言語の基礎を勉強した後、次のステップへ進むことを考慮したためです。最初は命令文の意味などにはとらわれずに、あくまでコンピュータへ命令するための〝記号〟として覚えていただければけっこうです。
BASIC言語をマスターするには、とにかく実際にキーボードに触れてプログラムを作ってみることです。昔から「習うより慣れろ」という言葉がありますが、BASIC I で BASIC言語に早く慣れてください。

このテキストは、小学校の高学年の方にでも理解できるように、言葉使い、例題などを選んでありますが、お父様、あるいはお母様とお子様がいっしょに勉強されるのも良いでしょう。本機をテレビに接続し、テキストの例題を実際に行ないながら説明文をお読みください。また、BASIC I でBASIC言語の基礎がマスターできたら次のステップとして、本格的なゲーム・グラフィック用BASIC言語「BASIC G」、あるいは実務計算までできる「BASIC F」へ進んでください。

1982年11月

株式会社 **タカラ**

はじめに

BASIC I を使う前に

Step 1 Basic I Ready

Step 2 BASIC I で何ができるかな？

Step 3 変数って何でしょう

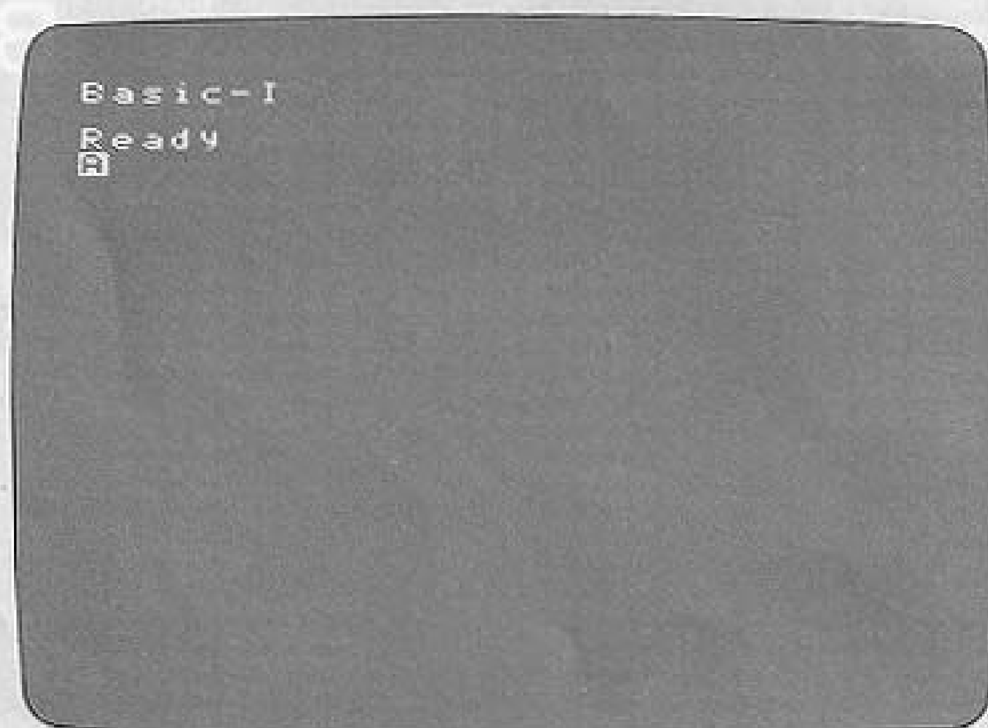
Step 4 プログラムって何？

Step 5 プログラムをとっておくには

Step 6 いろいろな命令語

1－1　スイッチ・オン

本体をテレビに接続し、BASIC Iのプログラム・カートリッジをセットしたら、電源ユニットのスイッチを“ON”にしましょう。（一瞬、画面にでたらめな表示が現われますが異常ではありません。）画面の左上に“BASIC I Ready”と、表示されていますか？
もし、この表示が出ないときは、すぐに電源スイッチを“OFF”にし、本体とテレビの接続、カートリッジの差し込みなどをチェックしてください。



この表示は、「これからBASIC Iが使えますよ。」と、コンピュータからあなたへの最初のメッセージなのです。

カーソル“A”が画面の左はしにかくれてしまうときは、次のようにしてください。
V I E W 1 , 0 , 3 0 , 2 3 RETURN
ただし、画面の左右に書ける文字数は1文字へって、GI、GII、マルチカラー・モードでは31文字、テキスト・モードでは39文字になります。

１－２　キーボード

さて、いよいよBASICⅠの指令で本機を動かしますが、キーボードのキーには色々な文字や記号が書いてありますね。これは、本機がひじょうにたくさんの機能を持っているからなのです。でも、基本的には下図の様な部分に区別することができます。

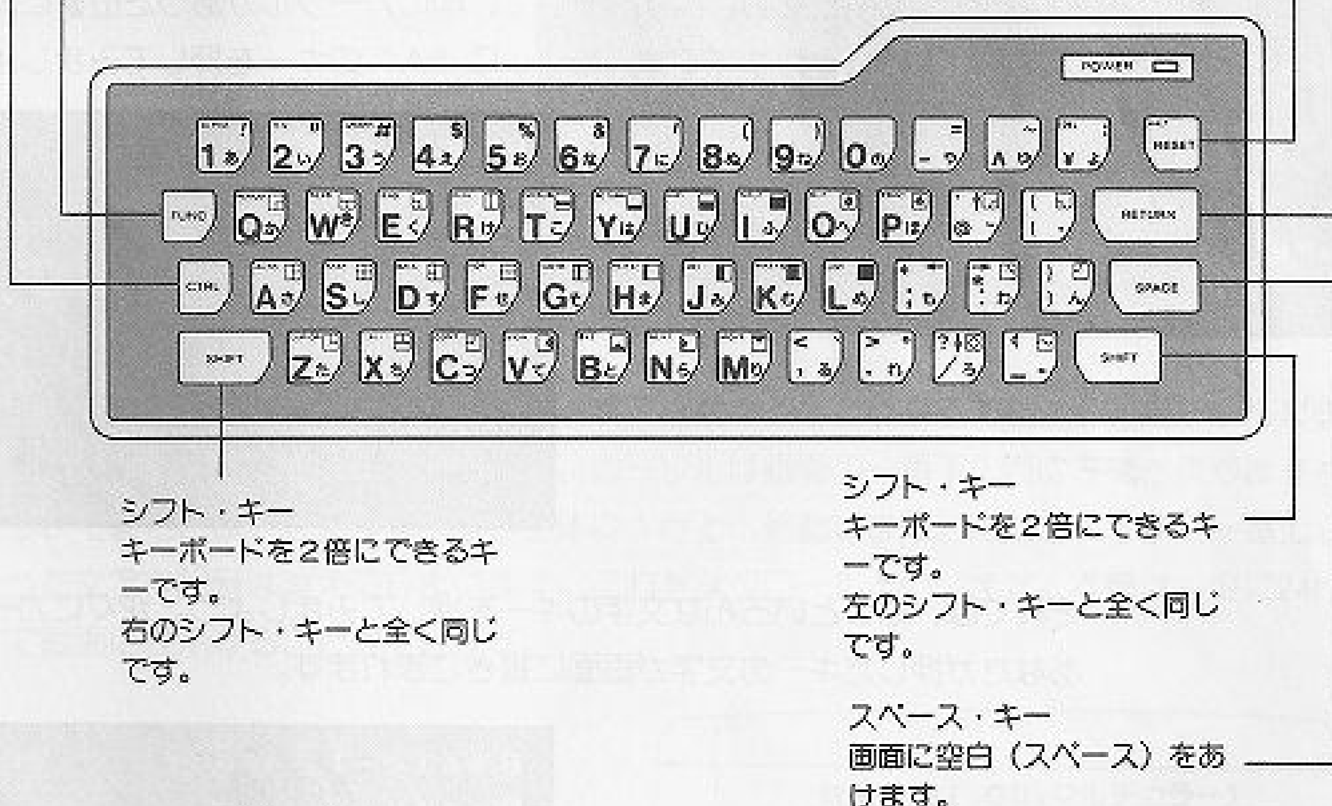
キーに書かれている文字や記号を画面に書くことができます。

コントロール・キー
キーボードの機能をコントロールできます。

フロンクション・キー
画面に書き込む文字や記号の種類を教えます。

リターン・キー
コンピュータに指令を送るためのキーです。

リセット・キー
終ったり・やり直したりするときに押しますが、あまり使わないようにしてください。



シフト・キー
キーボードを２倍にできるキーです。
右のシフト・キーと全く同じです。

シフト・キー
キーボードを２倍にできるキーです。
左のシフト・キーと全く同じです。

スペース・キー
画面に空白（スペース）をあけます。

くわしい使い方は、これから順番に説明していきます。

画面を見てください。"A"という表示が点滅していますね、これを"カーソル"といいます。

```
Basic-I

Ready
A
```

キーボードの文字のキーをどれかひとつ押してみてください。"ピッ"と音がして、カーソルは右へ1文字分移動します。そして、前にカーソルがあった位置にあなたが押したキーの文字が現われるのはずです。ためしに"A"のキーを押してみましょう。

```
Basic-I

Ready
aA
```

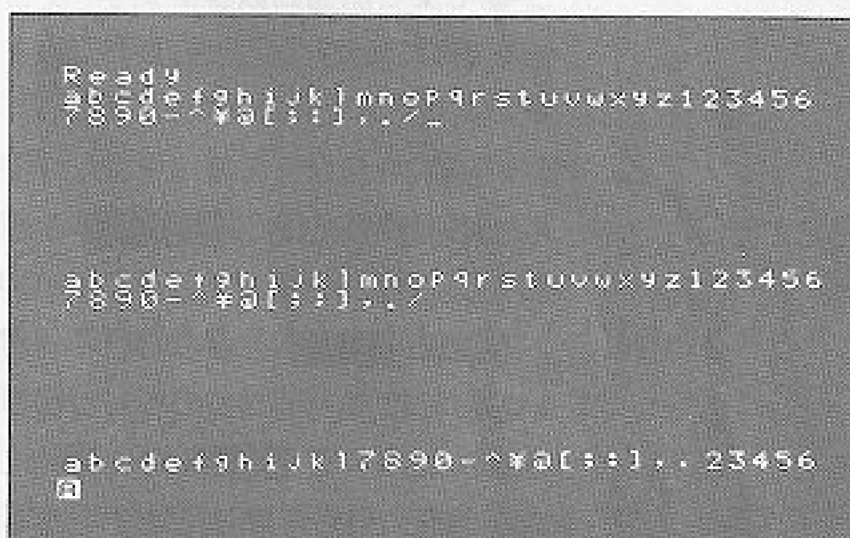
それでは、もっといろんな文字のキーを押してみましょう。次々にカーソルは右へ移動し、あなたが押したキーの文字が画面に書きこまれます。

```
Basic-I

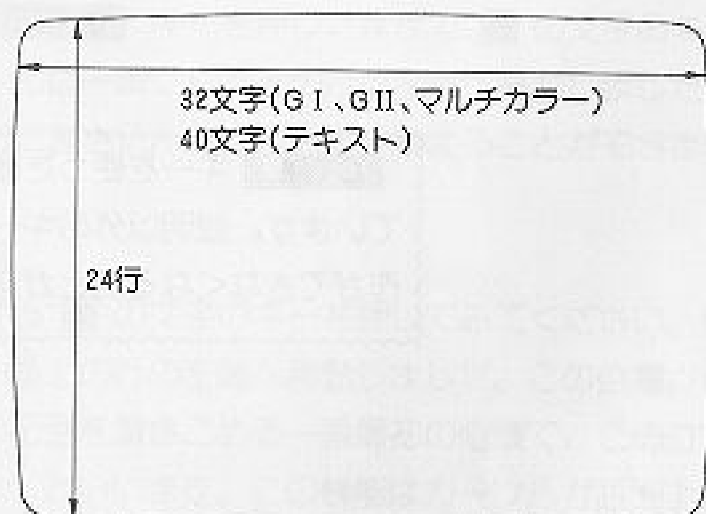
Ready
abcdefA
```

つまり"カーソル"は、「ここに書きこみますよ」と画面上の位置をあなたに知らせているのです。
先ほどの1－2で説明した、画面に文字や記号を書くことができるキーを全部押してみてください。

●スクロール機能
いかがですか、画面に文字や記号を書くことができるキーを全部押してみましたか？文字や数字、記号が次々に書きこまれ、カーソルは画面の右端まで移動すると自動的に下の行の一番左へ移動し、続けて文字や記号を書きこまれます。



また、カーソルが画面の一番下で右端まで移動するとどうなるでしょう。カーソルが画面の外へ出てしまうことはありません。カーソルは画面の一番下の行の左端へ戻りますが、今まで書きこまれた文字や記号が全体に1行上へ移動します。ただし、今まで一番上にあった文字や記号は消えてしまいます。これをスクロールといいます。本機は一度に24行まで画面に表示することができます。



●リピート機能

こんどは、キーボードの文字のキーをどれか1つ押し続けてみてください。いかがですか、同じ文字が自動的に連続して画面に書きこまれますね。キーから指をはなすと止まります。このようにキーを押し続けるとそのキーの文字が自動的に連続して書きこめる機能を〝リピート機能〟といいます。また、この機能は、文字や記号を書きこむだけでなく、次に説明する CTRL キーを使ったときなどにも機能します。

●デリート機能

さて、ここまでは画面に続けて文字や記号を書き込むだけでした。でも、実際にこれからBASIC I で画面に数字や文字、記号などを書きこんでいくときにまちがって違う文字や記号を書いてしまうこともあるでしょう。一度書いてしまった文字や記号を消して書き直したい。このようなときにはどうしたらいいでしょう。

キーボードの左側に CTRL と書かれたキーがあります。このキーをまず左手の指で押してください。離さずにそのまま押し続けておいてください。そうしたら、次にキーボード右上に DEL と書かれたキーがあります。このキーを右手の指で1回押してください。いかがですか、画面上のカーソルが1文字左へ戻りましたね。では続けて CTRL キーを押したままで DEL キーを数回押してみてください。カーソルが DEL キーを押した数分左へ戻り、今まで書きこまれた文字を消していきます。
それでは、CTRL キーから指を離して、どれか文字のキーを押してみてください。カーソルは今まで通り右へ移動し、今押したキーの文字が書きこまれますね。
この様に、一度書きこんだ文字や記号を消す機能を〝デリート機能〟といいます。この機能を使えば、もしまちがった文字や記号を書きこんでしまっても正しく書き直すことができます。

CTRL キーを使った機能には、ここで説明する機能以外にいろいろな機能を持っています。説明以外のキーを押すと、画面の文字が消えてしまったり、次の説明の操作ができなくなることがありますので、必ず説明通りに操作してください。

● エディット機能

では、もし画面にいろいろ文字や記号を書き込んだあとで、最初の方にまちがいを見つけたときはどうすればよいのでしょう。何行分もデリート機能で消して書き直すのでは大変ですね。

それでは、もう一度 CTRL キーを左手の指で押し続けてください。キーボードの右側に ←→↑↓ のマークが書かれたキーがありますね。まず ↑ キーを１回押してみてください。いかがですか、カーソルが左右の位置はそのままで１行上へ移動しました。

（ CTRL キーが押されていないと動きません。）

こんどは、 CTRL キーを押したままで ← キーを押してください。カーソルは同じ行で１文字左へ移動します。でも、カーソルが移動した場所の文字はそのまま残っていますね。

では、 CTRL キーから指を離して、文字のキーをどれか押してみてください。いかがですか、今まで書かれていた文字が新しく押したキーの文字に変わりました。

この様に CTRL キーを押しながら ←→↑↓ キーを押すことで、カーソルだけを画面上で自由に動かすことができます。そして、まちがって書きこんだ文字を直したり、新しく追加したりすることができます。この機能を〝エディット機能〟といいます。

いかがですか、これであなたはもう画面上に自由に文字や記号を書いたり、訂正したりすることができるようになったはずです。

それでは、ついでにもう少し CTRL キーを使った便利な機能を説明しましょう。

● カーソルを行の最初へ戻す機能

画面に文字や記号を書き込んでいる途中で、その行を全部書き直したい。このようなときには、先に説明した〝デリート機能〟や〝エディット機能〟を使って書き直すこともできますが、もうひとつ便利な機能があります。カーソルを、どこか文字が書き込まれている行の途中へ移動してください。そして、 CTRL キーを押したままで B の文字のキーを押してください。いかがですか、カーソルは一瞬にしてその行の左端へ移動しましたね。こうして、新しく文字を書きこめば、最初に書き込んだ文字を書きかえることができます。

● カーソルを画面の最初の位置へ戻す機能

さて、こんどは CTRL キーを押しながら K の文字のキーを押してみてください。いかがですか、こんどはカーソルが画面の一番上の行の左端へ移動しました。この位置が画面の最初の位置、つまり、画面上に文字や記号を書きこめる一番最初の位置で、この位置のことをカーソルの〝ホーム・ポジション〟といいます。この機能はカーソルが画面上のどこにあっても一瞬にしてカーソルをホーム・ポジションに戻すことができます。

●文字をまとめて一度に消す機能

カーソルをどこか文字が書きこまれている行の途中へ移動してください。そうしたら、もう一度 CTRL キーを押しながら、こんどは X の文字のキーを押してみてください。どうですか、カーソルがある行の、カーソルから右の文字が一度に全部消えてしまいました。

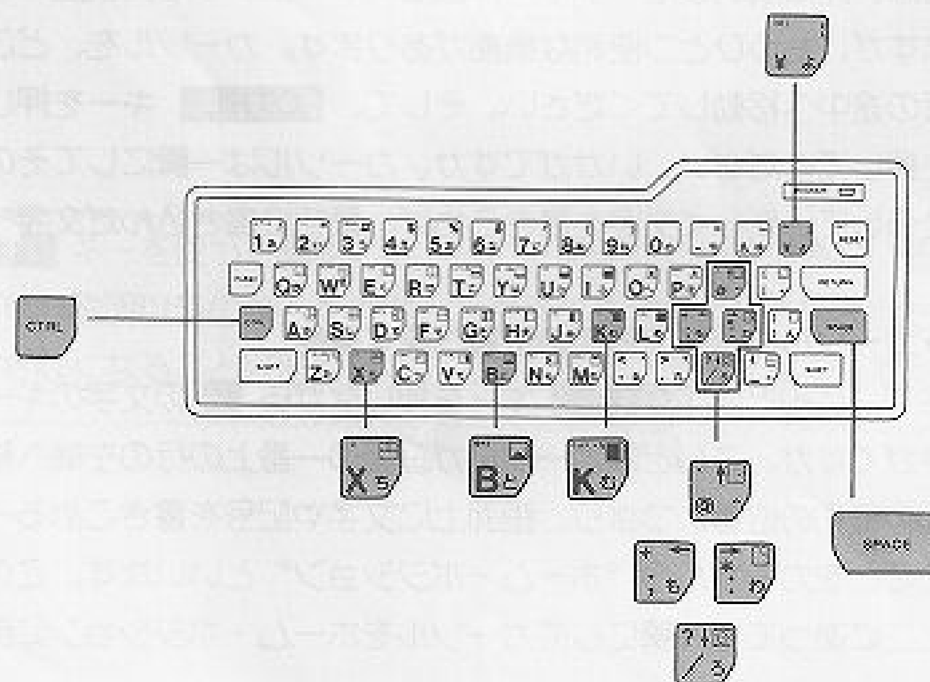
では、カーソルをもう一度どこか文字が書き込まれている行の左端へ移動して、CTRL キーを押しながら X キーを押してみてください。こんどはカーソルがある行全部が一度に消えてしまいましたね。このように、文字や記号をまとめて一度に消してしまうこともできます。

●デリート機能のもうひとつの使いかた

CTRL キーと X キーを使うと、カーソルのある位置から後の文字や記号を全部一度に消してしまいましたね。では、書きこんだ文字や記号の途中を何文字か取ってしまいたいときはどうすればよいでしょう。

まず、カーソルをどこか文字が書きこまれている行の途中へ移動してください。キーボード右端に SPACE と書かれたキーがあります。このキーを押してみてください。いかがですか、文字は消えましたが間があいてしまいました。このように、SPACE キーを使って文字を消すこともできますが、消した後は空白となって残ります。

では CTRL キーを押しながら DEL キーを1回押してください。いかがですか、カーソルの位置はそのままで右側の文字が1文字分左へつまりましたね。このようにすると、書きこんだ文字や記号の途中にいらない文字などが書かれてしまったときでも取ってしまうことができます。



1－4　画面にひらがなや図形を書くときは

さて、ここまででああなたは画面に自由に文字を書きこんだり、訂正したり、消したりすることができるようになられたと思います。ところで、もうあなたもお気付きになられていると思いますが、画面には、アルファベットの小文字と記号、及び数字しか書くことができませんでしたね。キーにはもっといろいろな文字や記号、図形なども書いてあります。どうしたらこれらの文字や記号、図形を画面に書くことができるのでしょうか。

●アルファベットの大文字を書くには

キーボードの右下と左下に SHIFT と書かれたキーがあります。ではこのキーをどちらかひとつ押したままでアルファベットのキーを押してみてください。いかがですか、画面にはアルファベットの大文字が書きこまれたはずです。それでは、 SHIFT キーを押したままで他のアルファベットの文字も書き込んでみてください。
では次に、 SHIFT キーを押したままでキーボードの一番上の列と、キーボード右側の数字や記号のキーを順番に押してみてください。どうですか、それぞれのキーの上側に白で書いてある記号が画面に書きこめましたね。
このように、 SHIFT キーは、ひとつのキーで２種類のちがう文字や記号を書くことができる、つまりキーボードを２倍にするキーなのです。そして、 SHIFT キーを使って画面に文字を書きこむことを〝シフトする〟といいます。

これでアルファベットの大文字、小文字、数字、記号の書き方がわかりました。このように、キーボードからアルファベットを書きこめる位置を〝アルファベット・ポジション〟といい、カーソルは〝 A 〟が表示されます。本体の電源を〝ON〟にしたとき、キーボードは必ずこのアルファベット・ポジションとなり、カーソルは〝 A 〟と表示されます。また、次に説明する〝ひらがな・ポジション〟や〝グラフィック・ポジション〟からこのアルファベット・ポジションに戻るときはキーボード左の FUNC キーと数字の 1 のキー〝ALPHA〟と小さな文字で書いてあるキーを同時に押します。

アルファベット・ポジションで書ける文字、記号

シフトキーを押さないとき。



シフトキーを押したとき。



●ひらがな・ポジション

それでは、キーボードに書いてある“ひらがな”を画面に書くのはどうすればいいのでしょう。キーボードの左端に FUNC と書かれたキーがあります。この FUNC キーを押しながら、キーボードの左上の数字の 2 のキーを押してください。数字の 2 のキーには左上に小さな文字で“かな”と書いてありますね。さて、カーソルを見てください、“A”から“K”に変わりました。それでは、あいうえおの順でキーを押してください。どうですか、画面にひらがなが書けますね。このように“ひらがな”を書きこめる位置を、“ひらがな・ポジション”といいます。また、このポジションでも SHIFT キーを使うことができます。

ひらがな・ポジションで書ける文字、記号

シフトキーを押さないとき。



シフトキーを押
したとき。

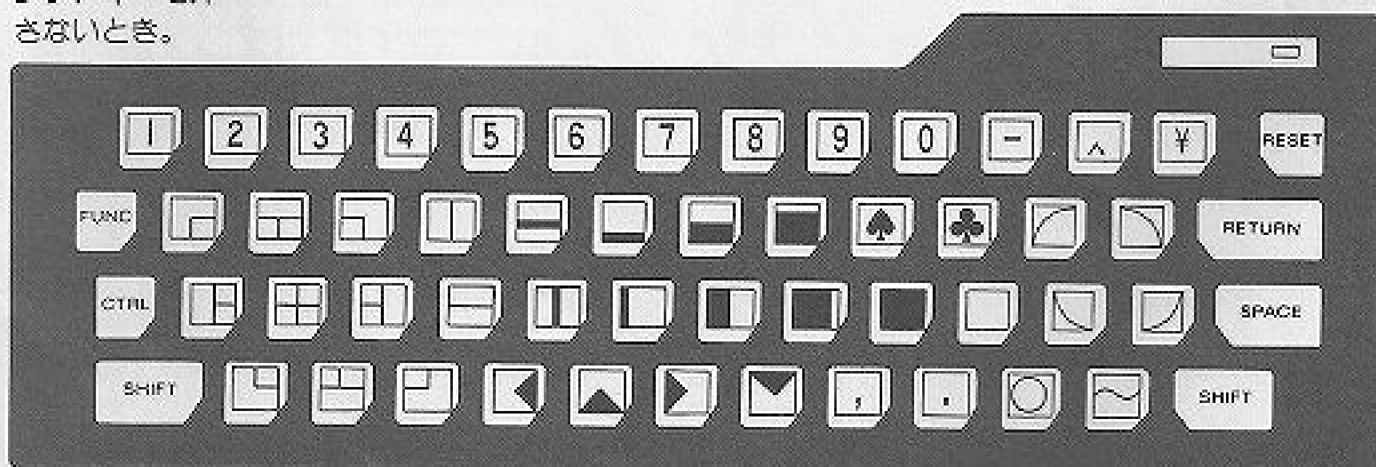


●グラフィック・ポジション

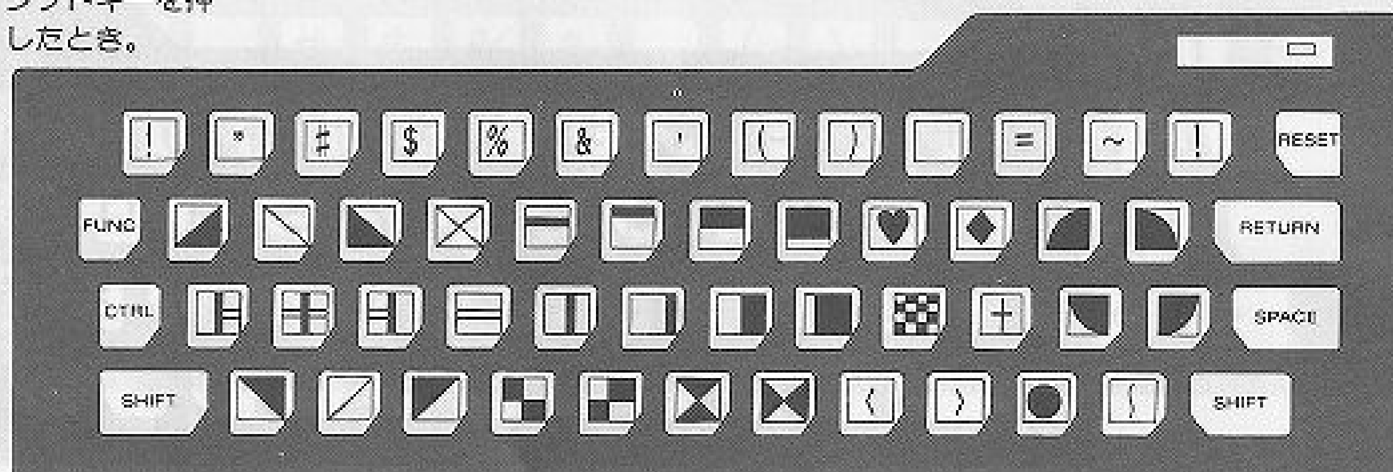
こんどは、キーボードのキーに青色で書いてある図形（グラフィック・パターン）を書いてみましょう。 FUNC キーと、こんどは数字の 3 のキーを同時に押してください。数字の 3 のキーの上に "GRAPH" と小さな文字が書いてあります。カーソルが" G "に変わりましたね。この位置を "グラフィック・ポジション" といい、キーに青色で書いてある図形を画面に書き込むことができます。このポジションでも SHIFT キーが使えます。実際にキーを押して画面に図形を書きこんでみてください。また、 CTRL キーと ←→↑↓ キーを使ってカーソルを移動させながら図形を組み合わせると、画面全体に表や、絵を書くこともできます。

グラフィック・ポジションで書ける図形、数字、記号

シフトキーを押
さないとき。



シフトキーを押
したとき。



キーボードの一番上の列の数字、記号は、アルファベット・ポジションと同じものが書きこめます。

さて、これでもうあなたは、画面にアルファベット、数字、記号、ひらがな、図形を自由自在に書き込めるようになりましたね。
このように、画面上に自由自在にカーソルを動かして、文字や図形などを書き込んだり、消したりすることは、BASIC I でプログラムを書いたり、プログラムを修正したりするときの基本になります。十分に練習してキーボードの操作になれてください。

Step 2

BASIC I で何ができるかな

PRINT

2－1 計算をやってみよう

※Step 1 から続けてお読みの方は、一度電源を切って画面をきれいにしてください。

さあ，いよいよこのステップでは BASIC Ｉ を使ってコンピュータに何かさせてみましょう。

●たし算をやってみよう

では，「２＋３ 」の計算をさせてみましょう。電卓ですと，2＋3＝ とキーを押せば答を出してくれますが，コンピュータに計算させるにはこれだけでは何もしてくれません。ためしに電卓と同じようにキーを押してみてください。画面に式が書きこまれるだけですね。BASIC Ｉ では、コンピュータに何かさせたいときには，ある決まった言葉でコンピュータに命令をしなければいけないのです。この，BASIC Ｉ で決められた言葉を「命令語(コマンド)」といい，いろんな命令語が用意されています。表示をさせるときは「PRINT 」という命令語を使います。

では，今画面に書きこんだ式を消して，カーソルを最初の位置まで戻したら，次の様に画面に書きこんでください。

PRINT2＋3

「PRINT 」の文字は，アルファベットの小文字でかまいません。＋ は SHIFT キーを押しながら ； のキーを押します。

```
Basic-I

Ready
Print2+3
```

あれ，まだ答えを出してくれませんね。カーソルが 3 の右で点滅しています。「＝」を押すのかな？ …ちがいます。＝ が無いからではありません。コンピュータは，まだ画面に何が書きこまれるのかな，とカーソルを点滅させて待っているのです。書きこみが終ったよ，という合図を送ってやらないと，コンピュータはいつまでも待っています。キーボー

ドの右端に RETURN と書かれたキーがありますね，このキーがコンピュータに書きこみが終ったことを合図するキーで，「リターン・キー 」といいます。では，このキーを押してみてください。

```
print 2+3
5 ←── 答えが出ました。
Ready
```

●ひき算，かけ算，わり算もできます。

最初に「PRINT」と書いて，次に算式を書いて，最後に RETURN キーを押してください。たし算と同様に答えが下の行に表示されます。

●ひき算の記号(−)は，数字の列の0の右にある − キーです。

●かけ算の記号(×)は，キーボード右側の * のキーです。 SHIFT キーを押したままで押してください。

●わり算の記号(÷)は，キーボード右下の / キーです。

(例)

7−3＝ P R I N T 7 − 3 RETURN

3×4＝ P R I N T 3 * 4 RETURN

6÷2＝ P R I N T 6 / 2 RETURN

いろいろと数を変えて計算をしてみてください。ただし，BASIC Ⅰ では，小数点以下の計算，及び最も小さな数が −32768 から，最も大きな数 32767 の間の数しか計算することができません。計算させる数や，答えがこの数の範囲を越えると計算できません。

(例)

456×789＝を計算させると図の様に表示されます。

```
Basic-I

Ready
Print456*789
Err 6 in 0
Ready
```

下の行の「Err 6 in 0」は計算の答えではありません。「計算できません」という意味のメッセージなのです。計算の答えは359784となるはずですが，BASIC Ⅰで計算できる数32767をオーバーしたため計算できないのです。

（例）

8 ÷ 5 = を計算させると，こんどは次の様に表示されます。

```
Basic-I

Ready
Print8/5
 1

Ready
```

下の行の1は答えです。本当の答えは「1.6 」ですね。でもBASIC Ⅰでは小数点以下の計算ができませんので「.6 」が無くなってしまったのです。

★エラー・メッセージとは

先ほどの説明でてできた，「計算できません」という意味のメッセージの「Err 6 in 0」，これを「エラー・メッセージ」といいます。

BASIC Ⅰでは，コンピュータへの命令が正しいかどうかをいつもチェックしています。もし，まちがった命令が書かれると，どこがまちがっているのかをErr 1〜Err 17のメッセージでおしえてくれるのです。それぞれのエラー・メッセージの意味は，*107*ページの「エラー・メッセージ一覧表」に書いてあります。もし，エラー・メッセージが表示されたら，この「エラー・メッセージ一覧表」を見て，まちがいを直してください。

●連続計算，カッコをつけた計算，るい乗（べき乗）の計算もできます

次の様な計算もできます。また，計算の優先順序もちゃんと守って計算できます。

（例）

12 × 5 − 36 ÷ 9 = [P][R][I][N][T][1][2][*][5][−][3][6][/][9][RETURN]

(2 + 4) × (6 − 3) = [P][R][I][N][T][(][2][+][4][)][*][(][6][−][3][)][RETURN]

$3 + 2^4$ = [3][+][2][∧][4][RETURN]

↑るい乗の記号は[∧]を使います。

※多重カッコの計算もできます。
計算の優先順序は，（　　）→るい乗→×または÷→＋または－の順となります。

いろんな式で計算をしてみましょう。（ただし，答えが小数点以下や，－32767～32767の範囲をこえると計算できません。）

いかがですか，PRINT という文字を使えば計算ができましたね。このPRINT の文字が，BASIC Ｉ の命令語のひとつで，ここでは，「次に書く計算式の答えを画面に表示しなさい」という意味に使っていたのです。このようにPRINT 命令を使うと，コンピュータが何か行った結果を画面に表示させることができます。そして，このPRINT 命令がBASIC 言語のもっとも基本的な命令語ともいえるものです。

2－2 PRINT命令で他に何ができるのかな？

●PRINT命令で画面に文字を書こう。

さて，ここまでの説明でPRINT命令を使って計算の答えを表示させることができました。では，先ほどのたし算の書き方をチョット変えて書いてみましょう。PRINT"2+3" RETURN と，計算式の前とうしろに "（ダブルコーテーションマーク）をつけてみました。いかがですか，答えが表示されずに計算式が表示されてしまいました。

```
Basic-I

Ready
Print"2+3"
2+3

Ready
```

このように，PRINTのあとの計算式を" "でかこむと，「" "でかこんだ部分をそのまま画面に表示しなさい」という意味になり，コンピュータは何も計算せずに，計算式をそのまま表示してしまうのです。

それでは，" "の中にいろんな文字や記号を書いて画面に表示させてみましょう。

```
PRINT "BASIC I"
PRINT "こんにちは！"
PRINT "♣♠♥♦"
```

いかがですか，RETURN キーを押すと，" "でかこんだ部分の文字や記号，図形が次の行に表示されますね。

実際にBASICで何かプログラムを作ったとして，このPRINT命令がなかったら，コンピュータはそのプログラムで動かすことはできますが，画面には何にもでてこないことになってしまいます。このため，PRINT命令は，実際のプログラムの中でも非常に多く使われる基本的な命令語なのです。

Step3

変数って何でしょう

3－1 変数とは？

「変数(へんすう)」，ちょっとむづかしいことばがでてきましたね。とりあえず説明はあとにして，次の様な方法でコンピュータに計算をさせてみましょう。

たとえば，15と９をたし算するのに「15＋９」という書き方のほかに次の様な方法もあります。つまり，15という数を「A」とします。次に，９という数を「B」とします。そうして，「A」と「B」をたし算する方法です。答えは，「15＋９」を直接行なっても，AとBにおきかえて「A＋B」としても同じですね。では実際にこの方法でコンピュータに計算させてみましょう。次の順番で画面に書きこみます。

1．A = 1 5 RETURN
2．B = 9 RETURN
3．P R I N T SPACE A + B RETURN

いかがですか，Step 2 で直接計算をさせたときと同様に答えが表示されましたね。
この方法で数のかわりにおきかえた「A」と「B」のことを「変数」といいます。もう少し正確にいうと，ここでの「A 」と「B 」は数をあらわしていますので，「数値変数(すうちへんすう)」といいます。

※変数は，数をおきかえるだけでなく，文字や記号についても同様におきかえることができます。このときの変数を文字変数といいます。文字変数については本書の後半でプログラム例とあわせて説明します。

●変数に使える文字の組み合わせ

いまの計算の例では変数として「A」と「B」を使いましたが，BASIC I では，どんな文字や記号を変数にしてもかまいません。しかも最大16文字まで組み合わせたものでも１つの変数として使えます。ただし，BASIC I の命令語と全く同じ文字の組合せと，最初の文字に数字を書くことはできません。たとえば，さきほどの例は次の様に書くこともできます。

```
りんご゛のねだ゛ん=100
みかんのねだ゛ん=200
Print りんご゛のねだ゛ん+みかんのねだ゛ん
```

```
♠=15
♣=9
Print ♠+♣
```

いかがですか，変数の意味が何となくわかっていただけましたか。このように，変数を使うということは，これからBASICのプログラムを作っていくうえでひじょうに大切な考え方なのです。ここでは，とりあえず変数とはこんなものなんだ，ということがわかっていただければいいでしょう。あとは，本書の後半でプログラムに実際に変数を使った例のところでもう一度説明します。

Step 4

プログラムって何？

CLS　NEW　RUN

LIST　INPUT　LET

AUTO　DEL

4－1 プログラムは，仕事の手順

さて，ここまでは，コンピュータに何かさせるのに，最初に命令語を書いて，次に計算式などを書いて RETURN キーを押すと，答えを出してくれました。しかし，もっと複雑なことをコンピュータにさせたいときにはこれでは大変です。そこで，コンピュータにさせたいことを番号を付けて順番に書いておき，全部書き終ったら最初から順番にコンピュータに実行させる。これがプログラムの考えかたです。つまり，コンピュータに仕事の手順をBASICのいろんな命令で覚えさせ，一度に実行させるものともいえます。

4－2 プログラムを作ってみよう

さて，いよいよプログラムにチャレンジしてみましょう。でも，その前に気分を一新する意味で今まで画面に書いてきた文字や数字，記号を全部消しましょう。電源を切るのではありませんよ。BASIC I の命令語を使ってやってみましょう。

ご注意

コンピュータの電源を“ＯＦＦ”にするということは，画面はもちろんきれいに消えますが，それと同時にコンピュータ本体のメモリなども全て消えてしまいます。もし，あなたがプログラムを作っている途中で電源を“ＯＦＦ”にしてしまったとしたら，それまでに作っていたプログラムは全部消えてしまいます。

●画面をきれいにする命令 CLS

この命令は，画面に書かれている文字や記号を全て消し，カーソルをホームポジションへ戻します。

まず，カーソルを画面で何も書かれていない行の先頭(左端)へ移動してください。

次に， C L S RETURN とキーを押します。

いかがですか，画面に書き込まれていた文字や記号等は全て消えてしまいました。今後，画面をきれいにするときは，この命令で行なってください。

●コンピュータの中をきれいにする命令 NEW

コンピュータの本体の中には，BASIC のプログラムを保管しておくためのメモリや，先ほどの変数をおぼえておくためのメモリなどがあります。もし，すでにプログラムが保管されているのに新しいプログラムを作ったとすると，２つのプログラムがごちゃごちゃになり，せっかくのプログラムが使えなくなったりすることがあります。そこで，新しくプログラムを作るときなど，この命令 NEW を使って，保管されているプログラムや変数の値などをあらかじめ全部消しておく必要があります。

N E W RETURN とキーを押します。画面には何の変化もあらわれませんが，これでコンピュータは，初めて電源を“ＯＮ”にしたときと同じ状態になりました。

●計算させるプログラムを書いてみよう。

次の様に画面に書きこんでください。

この３つの部分を合わせて、「プログラム文(ステートメント)」といいます。

行番号　　命令語　　オペランド

1 0 SPACE P R I N T SPACE 2 + 3 RETURN

2 0 SPACE E N D RETURN

いかがですか，先ほどのコンピュータで計算をさせるときの書き方とよく似ていますが，RETURN キーを押しても答えが表示されませんね。実は，それぞれの行の最初に書いた「10」，「20」の数字があるからなのです。先ほどの「プログラムって何？」のところで説明した，コンピュータにさせたいことに番号を付ける，つまり，この「10」，「20」の数字がその番号なのです。そして，このように行の最初に数字が書かれると，コンピュータは自動的に，「これはプログラム文だ」と判断し，プログラムを実行させる命令がくるまでは保管しておくだけで何もしないのです。また，この「10」，「20」の数字を「行番号」といいます。また命令語の後の「２＋３」の部分をオペランドといい，文番号－命令語－オペランドの組合せが標準的なプログラム文となります。それでは，行番号20（２行目）の「ＥＮＤ」は何でしょう。これは，「このプログラムはここで終りです。」とコンピュータに知らせる命令語なのです。プログラムの最後に必ず書きます。

●プログラムを実行させる命令 RUN

BASICのプログラムは，プログラムを実行させる命令がくるまで何もしません。この，プログラムを実行させる命令が RUN なのです。では， R U N RETURN とキーを押してみてください。いかがですか，答えが表示されましたね。ＲＵＮ命令が実行されると，コンピュータはプログラムを行番号の小さいものから順番に実行していきます。

※プログラムを実行させる命令 RUN を実行させることを，「プログラムをランさせる」とか，「プログラムを走らせる」などと言います。ＲＵＮ は英語で走るという意味だからでしょう。

```
10 Print2+3
20 end
run
 5

Ready
```

4－3 プログラムを見てみよう

●プログラムを見る命令 LIST

さて，前の説明では，プログラムを画面に書いて，そのまま RUN 命令で実行させました。でも，コンピュータはプログラムを保管しておくと説明しましたね。たとえば，画面上には最大で24行のプログラムしか表示できません。もし，24行以上のプログラムを書いたとすると，最初の方の行は画面上からは消されてしまいます。このようなプログラムを実行する前にチェックするときや，画面に書いたプログラムが正しくコンピュータに覚えられているかを確認するときに， LIST 命令を使います。

L I S T RETURN とキーを押してください。いかがですか，画面に先ほど実行させたプログラムと同じものが表示されましたね。このように LIST 命令を実行することを「リストを取る」といいます。

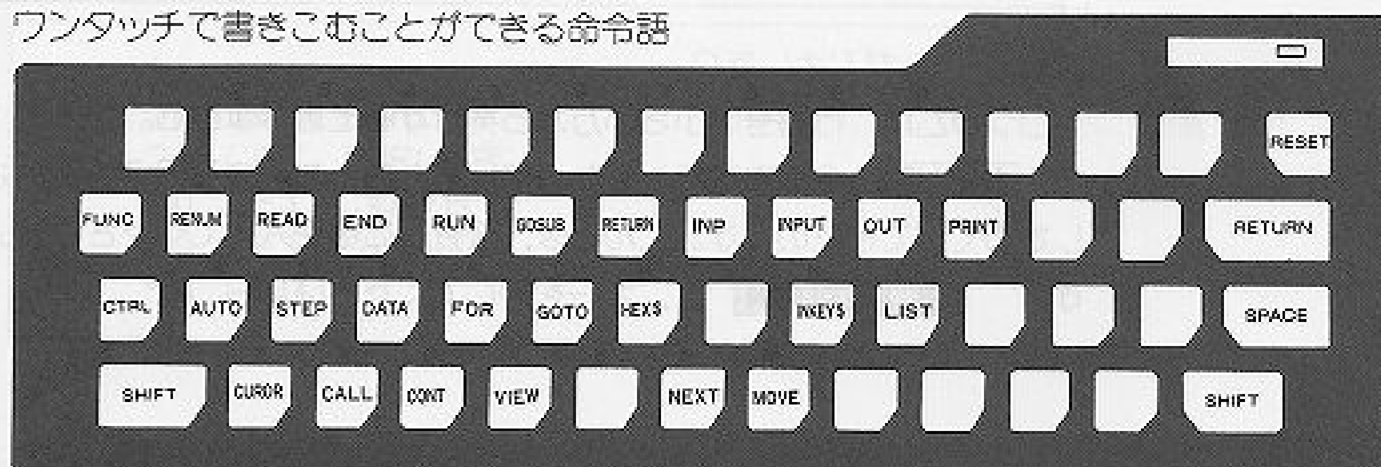
```
list
10 PRINT 2+3
20 END
```

※プログラム文の行番号と，命令語の間にスペースをとらなくてもちゃんとスペースが開いて表示されます。また，アルファベットの小文字は，大文字になっています。命令語とオペランドの間のスペースをこのプログラム文では書き込むときに開けなくても自動的に開けてくれますが，命令語の後にアルファベットがくるときには書きこむときに必ずスペースを開けなければいけません。

●便利な機能 FUNC キー

BASIC I の命令語がワンタッチで書きこめます。キーボードのキーの上側に小さな文字が書いてあります。その文字が画面にワンタッチで書き込めるのです。ただし，BASIC I にない命令語は書きこめません。たとえば，今までに説明にでてきた PRINT は，P の文字のキーにあります。キーボード左上の FUNC と書かれたキーを押しながら P を押してください。画面には，「PRINT」と書かれます。同様にして，他の命令語を書きこむことができます。

ワンタッチで書きこむことができる命令語



4－4 もう少しプログラムらしくしてみよう

●キーボードから数字や文字などのデータを入力させる命令

さきほどの計算をさせるプログラムは，わずか2行でした。これでもりっぱなBASIC のプログラムなのですが，「2 + 3」の計算だけしかできません。こんどは，途中で2つの数をキーボードから入力すると計算の答えを表示するプログラムを作ってみましょう。

まず，プログラムを画面に書き込むまえに，CLS ，NEW 命令を実行して，画面と前のプログラムを消してしまってください。

```
10 input a    ← スペースをあける
20 input b    ← スペースをあける
30 c=a+b
40 print c    ← スペースをあける
50 end
```

こんどのプログラムは，このようになります。画面に書きこんだら実行する前に，正しく書きこまれているかをよく確かめてください。

まちがいがなければ RUN 命令を実行させてください。

```
run
? ■
```

「?」（クエスチョンマーク）が表示され，その後にカーソルが点滅しています。これは，最初の INPUT 命令が実行されたためで，「変数Aに数を入れてください。」と要求しているのです。では，さきほどの計算と同じに「2＋3」を計算させてみましょう。

2 RETURN と，キーを押してください。

```
run
? 2
? ■
```

また，「?」がでましたね。

プログラムは，行番号の小さい方から実行すると書きました。初めの「?」は，行番号10の INPUT 命令を実行したために出てきました。2番目の「?」は，行番号20を実行したので出てきたのです。こんどは，変数Bに数を入れてください 。」と，要求しています。　　3 RETURN と，キーを押してください。

```
run
? 2
? 3
5 ←── 答えが出ました。
█
```

答えが出ましたね。

エッ，答えが出ない。それでは，次にリストを取って調べてみましょう。

●リストを取ってみよう。

ちゃんと実行した人も，エラーになってしまった人も，ためしにリストを取ってみましょう。

```
list
10 INPUT A
20 INPUT B
30 LET C=A+B
40 PRINT C
50 END
```

行番号30が違っているのでは？　あなたは行番号30では「30 c = a + b」と，書き込んだはずですが，ＬＥＴ(レット)という文字が入っています。

これは，LET 命令といって，この場合，「ＣにＡ＋Ｂの答えを代入しなさい」という意味になるのです。でも実は，この LET 命令は，省略されてもコンピュータはちゃんと理解できるのです。だから，リストを取ったときには，自動的に書きこんで表示されます。

●エラーがでてしまったら

答えが表示されずにエラーメッセージが出た人のリストは，たぶん次のようになっていると思います。

```
list
10 LET INPUTA
20 LET INPUTB
30 LET C=A+B
40 LET PRINTC
50 END
```

INPUT 命令，PRINT 命令の前に，LET 命令が入っていませんか。これは，書きこむときに行番号10で，

```
10 inputa
        ↑
        └── スペースを開けなかった
```

というように，INPUT 命令と変数「A」の間にスペースを開けなかったからなのです。
これは，「INPUT A」という命令文を，「INPUTA」という変数だと解釈してしまったからです。
このように，命令文の後に「変数」がくるときは，必ずスペースを開けるようにしましょう。
エラーメッセージが出た人は，正しいプログラムに直してから，もう一度実行させましょう。（プログラムの直し方は，33ページの4－5を見てください。）
このプログラムは実行する毎に数字を変えて計算させることができます。計算させる数をいろいろと変えて実行してみてください。

4－4 行番号・いろいろ

プログラムには，絶対に行番号が必要です。なぜ，行番号が必要なのでしょうか。

●行番号をつけないと，プログラムにならない。

プログラムは，行番号の小さい方から実行して行くと書きましたね。では，プログラムに行番号がなかったら……？　実行する手順が分らなくなってしまいます。

ですから，行番号を付けずに命令文をキー・インすると，コンピュータは「これは，プログラムじゃなくて，すぐに実行する命令だな」と，解釈してしまい，その命令文を覚えずにすぐに実行してしまうのです。

命令をすぐに実行することを，「ダイレクトモード」，プログラムとして覚えることを，「プログラムモード」といいます。

●なぜ10，20，30……と付けるの？

何度もいいますが，プログラムは行番号の小さい方から順に実行します。ですから，行番号は1，2，3……と付けても，ちゃんと実行してくれます。でも，普通は10，20，30…と10番おきに付けています。これは，後からプログラムの追加をするときに続けて行番号を付けてしまうと，いちいちプログラムを書き直さなければ追加ができなくなってしまうからです。ですから，実行する手順さえあっていれば，行番号は50番おきでも，100番おきに付けてもプログラムは，ちゃんと実行します。

● AUTO (オート)命令

行番号は必ず付けるものですから，ひとつの行番号の命令文が書き終ったら，自動的に次の行番号が出てくれば便利ですね。ところが AUTO 命令を使うとできるのです。

AUTO 命令は，「自動的に行番号を，発生させなさい」という命令です。たとえば，

①行番号100から10番おきにプログラムを作りたいときは，次のように書きこみます。ただし，10番おきの行番号自動発生のときは，最後の「，10」は省略できます。

```
AUTO 100,10
```

②行番号100から50番おきにプログラムを作りたいときには，次のように書きます。

```
AUTO 100,50
```

いろいろな行番号の自動発生ができますね。ただし，行番号に使える数は，1～32767までの整数です。

4−5 プログラムの直し方

プログラムの修正は，Step 1 で説明したキーを使ってできます。30ページの4−4でエラーが出たプログラムを，修正してみましょう。

```
10 LET INPUTA
20 LET INPUTB
30 LET C=A+B
40 LET PRINTC
50 END
█
```

● SPACE キーを使う。

①カーソルを，行番号10のLに重ねます。

② SPACE キーを押して，「LET」を消します。

```
list
10      █INPUTA
20 LET INPUTB
30 LET C=A+B
40 LET PRINTC
50 END
```

LET命令が消えました。

③カーソルをAに重ね SPACE キーを押します。

```
list
10      INPUT █
20 LET INPUTB
30 LET C=A+B
40 LET PRINTC
50 END
```

スペースをあける。

④ A の文字のキーを押します。

⑤ RETURN キーを押します。

これで，行番号10が修正されました。ここで注意しなければいけないのは，最後の RETURN キーを押し忘れると，プログラムはもとのままで修正されません。リストを取って確かめましょう。

● リストのテクニック

プログラムの修正の途中でリストを取りたいときは、カーソルを最後の行番号の下まで移動しなくても取れます。

CTRL キーを押しながら X キーを押し行番号20を全部消してしまいます。ここへ L I S T と書きこみ RETURN キーを押してください。いかがですか、今消したはずの行番号20もちゃんと出てきます。CTRL X では画面の表示を消しただけなのです。

```
10 INPUT A
20 INPUT B
30 LET C=A+B
40 LET PRINTC
50 END
```

行番号10は修正されました。行番号20も消えていませんね。

● DEL キーを使う。

DEL キーを使っても修正できました。行番号20を、Step1-3を思い出して、自分で修正してみましょう。

行番号20は直りましたか。リストを取って確かめてください。

● 1 行全部を書き直す。

まだ行番号40がまちがっています。いま、カーソルのある位置からこんどは行番号40の先頭にカーソルを移動し、そのまま正しいプログラムを書き込んでください。

```
40 PRINT C
        └── スペースをあける
```

これで、行番号40は書き変えられました。リストを取ってみましょう。全て直りましたね。

```
10 INPUT A
20 INPUT B
30 LET C=A+B
40 PRINT C
50 END
```

このように、同じ行番号に書かれた命令文は、後から書いた方が優先されます。

● 1行全部を消してしまうには、

行番号50を消してみましょう。行番号だけを書き、 RETURN キーを押します。
これで、行番号50は消えてしまいました。

```
50    RETURN キーを押す
  └─ なにも書かない
```

● DEL (デリート)命令って？

NEW 命令では、全部のプログラムが消えてしまいましたね。そこで不必要なプログラムだけを消すのが、DEL 命令です。
せっかく直したプログラムですが、DEL命令を使って消してみましょう。行番号20から行番号40を消してみます。DEL のあとに消したい 最初の行番号 , 最後の行番号 を書きます。

D E L 2 0 , 4 0 RETURN

これで、行番号20から行番号40が消えてしまいました。リストを取ってみましょう。

```
list
10 INPUT A
Ready
```

確かに消えています。
DEL命令は、連続した不必要なプログラム文を消すときに便利です。

4－6　こんな使い方もできます・INPUT 命令

こんどは、次の様なプログラムを実行してみましょう。CLS，NEW 指令で画面と前のプログラムを消したら書きこんでください。

```
new
10 input a,b
20 c=a+b
30 print c
40 end
```

4－3のときのプログラム

```
10 INPUT A
20 INPUT B
30 LET C=A+B
40 PRINT C
50 END
```

4－3のときとは少し違うプログラムのようですが、結果は全く同じになります。つまり、行番号10と20の INPUT 命令を、行番号10でまとめて書いてあるのです。
実行してみましょう。

```
run
?
```

7＋5を計算させましょう。4－3のときのプログラムとは、キー・インのし方が違います。7 , 5 RETURN とします。

```
run
? 7,5      ← カンマ
12         ← 答えです
```

12と答えが出ました。
こうすると、たくさんの数をキー・インするとき便利ですね。
では、プログラムを、次の様にすこし変えてみましょう。

```
new
10 input a,b,c
20 d=a+b+c
30 print d
40 end
```

今度は、3つの数をキー・インして、それらを全部たし合せなさい、というプログラムです。
実行してみましょう。

```
run
?
```

３＋４＋５の計算をやってみましょう。

```
run
? 3,4,5
12 ←―― 答え
```

ちゃんと答えがでましたね。

● INPUT 命令のエラー・メッセージ

もう一度今のプログラムを実行して、「３＋４＋５」の計算をやってみましょう。

```
run
?
```

ここで、数字の変わりに文字を書きこんでみます。

```
run
? cd
-?
```

「－？」と表示されましたね。これは、『Aに代入すべきものは数字です、文字が書きこまれました。ちゃんと数字をキー・インしてください』と、いっているのです。
今度は「３、４、５」とキー・インする変わりに、「３、４」だけ書きこんで
キーを押してみましょう。

```
run
?  cd
-? 3,4
??
```

こんどは、「？？」が出ました。これは、『２つの数しか入力されていません、３つの数全部を書きこんでください』と、表示しているのです。そこで、入力し忘れた「５」だけを書きこみます。 INPUT 命令ではいろんなメッセージが出ますね。下にそれをまとめておきます。

－？……数字が入力されなければならないのに、文字や記号が入力された。

？？……入力すべき数字や文字が不足している。

（このときは、入力し忘れた数字や文字だけを後から入力する。）

何も入力せずに RETURN キーが押された。

4－7　メッセージでもっとわかりやすく

今までのプログラムでは、実行すると「？」が表示されるだけでした。これでは、プログラムを作った人にしか何を入力すればいいのかわかりませんね。そこで、今度は誰が見ても何を入力したらいいかわかるように「メッセージ」をいっしょに表示させるプログラムを作ってみましょう。

●どうしたらメッセージが出せるの？

CLS 指令で画面をきれいにしたら、先ほどのプログラムのリストを取ってください。まず行番号10を次のように書き換えます。

ダブルコーテーションマーク

```
10 INPUT "みっつの␣かず゛の␣たしざ゛ん␣a+b+c";a,b,c
```

└スペース

ひらがなは、キーボード左上の FUNC 2 で書けましたね。
さて、行番号10を書き変えたら、実行してみましょう。

```
run
みっつの かず゛の たしざ゛ん a+b+c
```

いかがですか、INPUT 命令のうしろに " でかこんだ部分が画面に表示されましたね。だいぶ分りやすくなりました。3＋4＋5を計算させましょう。カンのいい人なら、3，4，5と3つの数を入力すればいいとわかるはずです。

●もっとわかりやすく

それでは、もっとわかりやすくしてみましょう。できれば、a＝?、b＝?、c＝?と順々に出てくるようにしたいですね。では、今のプログラムを次の様に修正してみてください。

```
10 print "みっつの かず゛の たしざ゛ん a+b+c"
12 input "a=";a
14 input "b=";b
16 input␣"c=";c
```

スペースを忘れずに　　セミコロンです。これも忘れないように

行番号10と20の間に3行のプログラム文を追加します。
10番おきに行番号を書いておいたので、プログラムの追加が簡単にできますね。
それでは実行してみましょう。

```
run
みっつの かずの たしざん a+b+c
a=?
```

このプログラムだと絶対に、a，b，cの3つの数のたし算をするプログラムだということがわかりますね。しかも、「aに入れる数は何ですか？」と、メッセージまで出ます。
もう一度、3＋4＋5を計算させてみましょう。

```
run
みっつの かずの たしざん a+b+c
a=? 3
b=? 4
c=? 5
12
```

順番に「b＝?」，「c＝?」と聞いてくれます。コンピュータと問答をやっているようですね。

LIST命令・いろいろ

プログラムを実行させたとき、もし次のようなエラーメッセージが出たとします。

```
run
Err 2 in 30
```

「行番号30の書き方がまちがっていますよ。」と、いっています。
そこで、あなたは、リストを取ってまちがっている所を直すわけですが、この場合必要なのは、行番号30だけですね。プログラム全部を表示させてもよいのですが LIST 命令には次の様な便利な使い方があります。

①行番号30だけを表示する

```
list 30, 30
       ↑
       └────カンマ
```

②初めから行番号30までを表示する

```
list ,30
```

③行番号30から最後まで表示する

```
list 30
```

④行番号30から60まで表示する

```
list 30,60
```

いろいろなリストの取り方がありますね。しつこいようですが、RETURN キーを忘れずに。
また、たくさんのプログラムを入れた時、始めから終りまでのリストを見ようとすると、初めの方のプログラムは、どんどんスクロールされて画面から消えてしまいます。もし、途中でどうしても止めたいときは、SHIFT キーと RESET キーを同時に押してください。

Step 5

プログラムをとっておくには

SAVE　VERIFY　OLD

5－1　プログラムをセーブする

あなたが、せっかく苦労して作ったプログラムも、本体の電源スイッチを切ると一瞬にして消えてしまいます。
とっておきたいプログラムでは、こまりますね。でも、安心してください。プログラムはカセット・テープに録音してとっておけます。それを、プログラムの「セーブ」といいます。
Step4－7でやったプログラムをセーブしてみましょう。
次のようなプログラムでした。

```
10 print "みっつの かず の たしざん  a+b+c"
12 input "a=";a
14 input "b=";b
16 input "c=";c
```

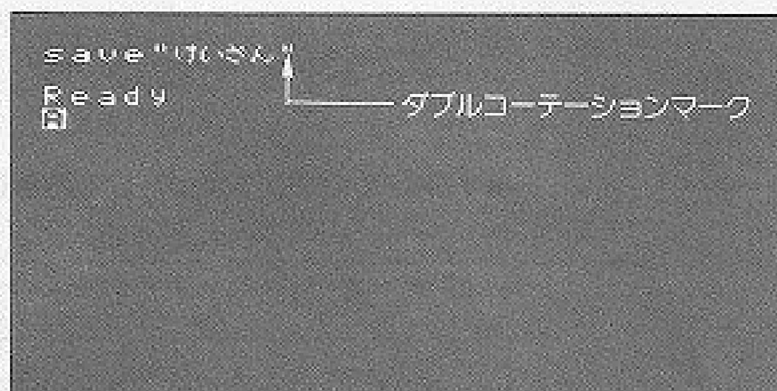
※プログラムをセーブする前に、カセット・テープレコーダを接続しましょう。接続方法は、ハンドブックの10ページを見てください。

●プログラムに名前をつけよう

プログラムをセーブするときには、そのプログラムに必ず名前を付けなければいけません。その名前を「ファイル名」といいます。ファイル名が付いていないと、そのプログラムはセーブされません。
それでは、先ほどの「3つの数のたし算」のプログラムをセーブさせてみましょう。ファイル名は“けいさん”にします。ここでつけたファイル名がカセット・テープにプログラムといっしょにセーブされます。次のように入力します。

```
save"けいさん"
```

RETURN キーを押すまえに、カセットを録音状態にします。リモート端子のあるカセットでは、まだテープは回らないはずです。
RETURN キーを押してみましょう。「カチッ」といってテープが回り始めます。リモート端子のないカセット・テープレコーダを使うときは「カチッ」と音がしてから、録音状態にします。しばらくすると、「カチッ」といってテープが止まります。画面を見てください。Readyと表示され、プログラムのセーブが終ったことを示しています。



リモート端子のないカセットを使うときは、セーブが終ったらテープを止めるようにしましょう。

●ファイル名について

ファイル名は、あとで何のプログラムなのかわかりやすくするためにそのプログラム内容にあった名前を付けるようにしましょう。そして、必ずファイル名とプログラムの内容をノートにひかえるか、カセット・テープのラベルに書いておいてください。音楽を録音したときに、曲名を書いておくのと同じですね。
また、ファイル名に使える文字数は、9文字までです。10文字以上のファイル名を付けると、プログラムはセーブされません。

5－2 ちゃんとセーブできたかな？

画面に Ready と出れば、プログラムのセーブは完了なのですが、ほんとうにセーブされたかチョット心配ですね。

●VERIFY（ベリファイ）してみましょう。

ほんとうにセーブされたか調べるときには、VELIFY（ベリファイ）命令を使います。さっきのプログラムのファイル名は、〝けいさん〟でしたね。次のように入力してください。

```
verify "けいさん"
```

RETURN キーを押すまえに，カセット・テープは，さきほどセーブを始めたところまで，テープを巻き戻しておきます。 RETURN キーを押したら，すぐにテープの再生ボタンを押してください。

```
verify "けいさん"
けいさん*  ←――――― ベリファイしていますよという印
```

VELIFY 命令で，コンピュータは自分で覚えている〝けいさん〟のプログラムと，カセットにセーブしたプログラムとを照し合わせて同じかどうか，調べています。しばらくすると，「カチッ」と音がしてテープは止まり，ちゃんとセーブされていると，「Ready」と表示されます。

```
verify "けいさん"
けいさん*
Ready
```

もしカセット・テープにうまくセーブされていないときは，エラーメッセージが出ます。そのときは，もう一度セーブをやり直すことになります。

※カセット・テープレコーダのボリウムのレベルが小さくてベリファイできないときもありますので，もう一度カセット・レコーダのボリウムを調整して，ベリファイし直してみてください。

※〝ファイル名〟を省略してベリファイしたときは，最初に見つけたプログラムをベリファイします。

●VERIFY のもうひとつの使い方

カセット・テープに入れたファイル名がわからなくなってしまったときなど，そのカセット・テープを始めから再生して，ベリファイしていけば，そのカセット・テープに入っているプログラムのファイル名を表示させることができます。ただし，このときでも〝ファイル名〟をつけておかないと最初のプログラムだけしか表示しません。また〝ファイル名〟

は，わざと全く使われていそうもない〝ファイル名〟にしておきます。さっきの〝けいさん〟のプログラムがセーブされているテープで，ためしてみます。
次の様に入力してください。

```
verify "たこ" ←—— 使われていそうもないファイル名をつける
```

テープを再生して RETURN キーを押します。しばらくすると画面に〝けいさん〟とファイル名が表示されます。

```
verify "たこ"
けいさん
```

カセット・テープにたくさんプログラムが入っているときは，〝たこ〟というファイル名をもつプログラムをさがしながら，ファイル名を次々と表示していきます。ただし，この場合カセット・テープが最後まで再生されてもベリファイの状態が続きます。止めるときは SHIFT キーと RESET キーを同時に押してください。

5－3 プログラムを読みこむ

カセット・テープにセーブしたプログラムを，コンピュータに読みこませるには，OLD（オールド）命令を使います。これを，プログラムをロードするといいます。

●プログラムのロード

プログラムをロードしたいときは，OLD 命令の後にロードしたいプログラムの"ファイル名"を付けます。
それでは今セーブした"けいさん"のプログラムをロードしてみましょう。
その前に，ほんとうにロードされたかどうか確かめるために，NEW 命令でいまコンピュータに入っているプログラムを全て消しておいてください。次の様に入力します。

```
old "けいさん"
```

RETURN キーを押してから，テープを再生してください。しばらくすると画面に次の様に表示されます。
※カセット・テープの巻き戻しを忘れないでください。

```
old "けいさん"
けいさん*←——— アスタリスク
```

ファイル名の後に付いている＊は，このプログラムをロードしていることを示しています。
たとえば，"けいさん"のプログラムの前に，"でんわばんごう"とか，"さいころげーむ"とかのプログラムが入っているとき，画面の表示は次のようになります。

```
old "けいさん"
て゛んわは゛んこ゛う
さいころけ゛ーむ
けいさん*
```

ファイル名の後に＊が付いていない"でんわばんごう"や"さいころげーむ"などのプログラムは，ロードされません。

●ファイル名を付けないと，

OLD 命令は，ファイル名を付けなくてもエラーになりません。ただし，そのときにロードされるプログラムは，OLD 命令が実行されてから，最初に出会ったプログラムです。さっきのように，"でんわばんごう"，"さいころげーむ"，"けいさん"の順でセーブされているテープを，始めから再生したときは，次の様に表示されます。

```
て゛んわは゛んこ゛う*
old
    ↑
    └──── 最初に出会ったプログラムをロードされる。
```

Step 6

いろいろな命令語

REM　GOTO　NEXT

FOR～TO～STEP

IF～THEN～ELSE

6—1 あきるまで計算させよう

Step 4〜7 でやった "けいさん" のプログラムがテープに入っている人は，ロードしてください。
テープに入っていない人は，次のようにに書きこんでください。

```
10 PRINT "みっつの かず の たしざ ん a+b+c"
12 INPUT "a=";A
14 INPUT "b=";B
16 INPUT "c=";C
20 LET D=A+B+C
30 PRINT D
40 END
```

これに，次のプログラムを追加してください。

```
2 REM けいさん
5 CLS
```

行番号2は，REM（リマーク）命令といって，この後にコメントを書くことができます。何を書いてもプログラムの実行には無関係です。ここでファイル名をコメントとして書いておくと，何のプログラムかがすぐにわかり，便利ですね。
行番号5は，CLS 命令で，画面をきれいにして，カーソルをホーム・ポジションへ戻す命令でした。

それでは，もう一行次の様なプログラム文を追加してください。

```
35 goto 10
        └── スペースをあける
```

それでは，いま新しいプログラム文を追加した「3つの数のたし算」を実行してみましょう。

いかがですか，今までは RUN 命令を書き込んだ下の行にメッセージが表示されていましたが，今度は行番号5が最初に実行されるために，一度画面がきれいに消され，画面の左上にメッセージがでます。
さきほどと同様に適当に計算をしてください。
どうですか，答えが表示されると，また初めのメッセージが出てきますね。どんどん計算してみてください。

```
RUN
みっつの かす゛の たしさ゛ん a+b+c
a=? 3
b=? 4
c=? 5
 12
みっつの かす゛の たしさ゛ん a+b+c
a=? █
```

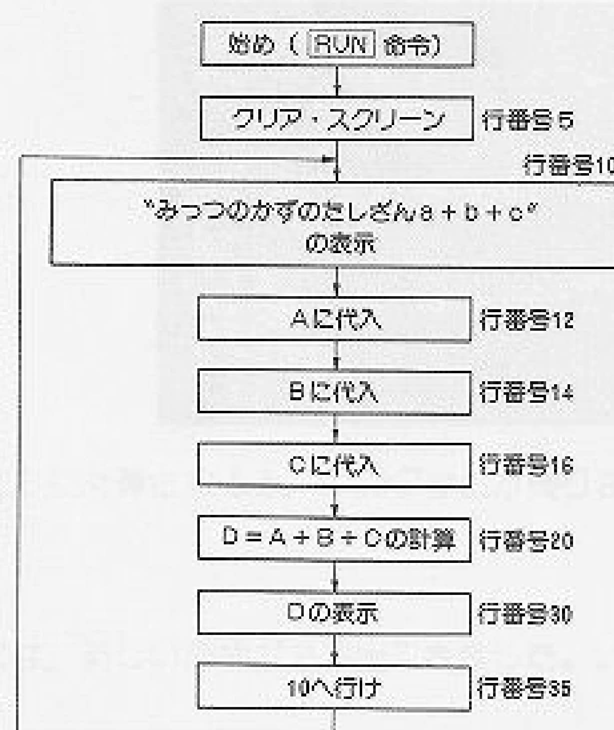
何度でもくり返し計算させることができますね。これは，行番号35が実行されているからなのです。

```
35 GOTO 10
```

という命令文は，「プログラムの実行が行番号35まできたら，そこから行番号10へ戻りなさい。」ということです。

これを，GOTO（ゴートウー）命令といい，GOTO 命令のあとに書いた行番号へプログラムの実行手順をジャンプさせる命令です。

プログラムは，行番号の小さい方から実行していくことは前にも書きましたね。このプログラムの実行手順を図で書いてみましょう。



このように，行番号35まで実行すると，［GOTO］命令により実行手順が行番号10へ戻されます。するとまた，行番号10，12，14，16，20，30と順に実行され行番号が35までくると，また［GOTO］命令で行番号10へ戻されます。いつまでたっても終りませんね。このように同じプログラムをくり返し実行することを「ループ」といいます。
また，ここでは限りなくプログラムを実行するので 特に「無限ループ」といいます。
そろそろ，計算させるのもあきてきました。プログラムを止めるのは［SHIFT］［RESET］でしたね。

6－2 3回だけ計算させよう

前のような「無限ループ」ではなく，こんどは何回計算させるか，数を限ってみましょう。次のようにプログラムを変えます。

```
2 REM
5 CLS
7 FOR Z=1 TO 3 STEP 1
10 PRINT "みっつの かず゛の たしざ゛ん a+b+c"
12 INPUT "a=";A
14 INPUT "b=";B
16 INPUT "c=";C
20 LET D=A+B+C
30 PRINT D
35 NEXT Z
40 END
```

前のプログラムのリストを取り，下の行に次のように書きこめば上のプログラムになりますね。

```
              ┌── スペースをあける
7 FOR␣Z=1 TO 3 STEP 1
35 NEXT␣Z
        └── スペースをあける
```

それでは実行させてみましょう。前と同じようにメッセージが出てきます。計算してください。

```
Ready
RUN
みっつの かず゛の たしざ゛ん a+b+c
a=? 2
b=? 3
c=? 4
 9
みっつの かず゛の たしざ゛ん a+b+c
a=? 3
b=? 4
c=? 5
 12
みっつの かず゛の たしざ゛ん a+b+c
a=? 5
b=? 6
c=? 7
 18
```

いかがですか，ちゃんと3回計算されると，プログラムが終りました。

●また新しい命令だ

さて，今のプログラムには，新しい命令が2つ出てきました。いやにならずに説明を読んでください。

まず行番号7から。これは次のようなプログラム文でしたね。

```
7 FOR Z=1  TO 3  STEP 1
```

変数Zに1から　3まで代入する　1つづつ

これは,「変数Zが3になるまでプログラムを1回実行する毎に1を加えて行きなさい」という命令で, FOR 命令といいます。
次は行番号35です。

```
35 NEXT Z
```

これは, 行番号7の FOR 命令と関係していて,「Zが3になるまで, FOR 命令のある行番号に行きなさい」という命令です。

この FOR 命令と NEXT 命令は, 必ずペアーになって使われます。どちらか1つだけを使うとエラーになります。
また, FOR 命令で使ったのと同じ変数(ここではZ)を, NEXT 命令で使わなければいけません。
FOR 命令の最後に出てきた,「STEP 1」というのは,「1を加える」と書きましたが正確にいうと,「連続して代入する」ということです。
たとえば, 3つおきに代入するようなときは,「STEP 3」とします。
このプログラムのように「1つずつ代入する」ようなときは, 省略できます。

6－3 FOR ～ TO ～ NEXT 命令

前に FOR 命令と NEXT 命令は，必ず対になって使われるといいましたね。ですから，これらの命令をまとめて FOR ～ TO ～ NEXT 命令とも呼びます。

FOR ～ TO ～ NEXT 命令を，もうすこし見てみましょう。

こんどは次のようなプログラムを書きこんでください。前のプログラムは使いませんので NEW 命令を実行してから書きこんでください。

```
10 CLS
20 FOR A=1 TO 10
30 PRINT "iwasaki"
40 NEXT A
```

（"iwasaki" ← あなたの名前を書きましょう。）

これは，あなたの名前を10個書かせるプログラムです。実行してみましょう。

アッという間に，あなたの名前が10個書けました。

50個書かせたいときは？　もう分りますね。行番号20を次のように修正します。

```
20 FOR A=1 TO 50
```

● 1 ～50まで数えささよう。

こんどは，1 ～50まで数えさせてみましょう。今のプログラムの行番号30を次のように書き直します。

```
30 PRINT A
```

実行してみましょう。1 ～50まで書けました。

```
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
Ready
```

なぜ，1 ～50まで順番に書けるのでしょうか。リストを取ってください。

```
10 CLS
20 FOR A=1 TO 50
30 PRINT A
40 NEXT A
```

（行番号20～40：ループです）

行番号20と40のFOR～TO～NEXTループは，Aが50になるまでくり返えされるはずでした。そのループの中に PRNT 命令「Aの値を表示しなさい。」が，書かれています。FOR～TO～NEXT 命令の中にも同じ変数Aが使われていますね。
だんだん分ってきましたか。変数Aには，ループが1回くりかえされるごとに，1，2，3，4，……50と数値が代入されています。そのループの中に行番号30で「Aの値を表示しなさい。」という命令がありますから，ループがくり返えされる毎に，1，2，3，4，……50と表示されるのです。
あたかも，1～50まで数えているようですね。

● 5つおきに数えさせる。
前のプログラムは，1から順番に1つずつふやして50までくり返しました。
こんどは5つおきに数えさせてみましょう。行番号20を次のように書き直します。

```
20 FOR A=1 TO 50 STEP 5
```
（STEP 5 ← 追加します）

前のプログラムは，1つずつ数えさせていたのでSTEPは，省略していました。こんどのプログラムは，5つおきに数えさせますから「STEP 5」とします。実行してみましょう。

```
 1
 6
 11
 16
 21
 26
 31
 36
 41
 46
Ready
```

ちゃんと5つおきに数えていますね。数を数えることを「カウントする。」ともいい，数を数えるプログラムを「カウンター」ともいいます。
それでは， FOR～TO～NEXT 命令を使って，いろいろなカウンターを作ってみましょう。

○ －100から100まで1つずつカウントして表示する。

```
10 FOR A=-100 TO 100 STEP 1
20 PRINT A
30 NEXT A
```
（STEP 1 ← 省略してもよい）

○ 200から0まで1つずつカウントして表示する。

```
10 FOR A=200 TO 0 STEP-1
20 PRINT A            └── マイナス
30 NEXT A
```

○ 100から－200まで10ずつカウントして表示する。

```
10 FOR A=100 TO-200 STEP-10
20 PRINT A
30 NEXT A
```

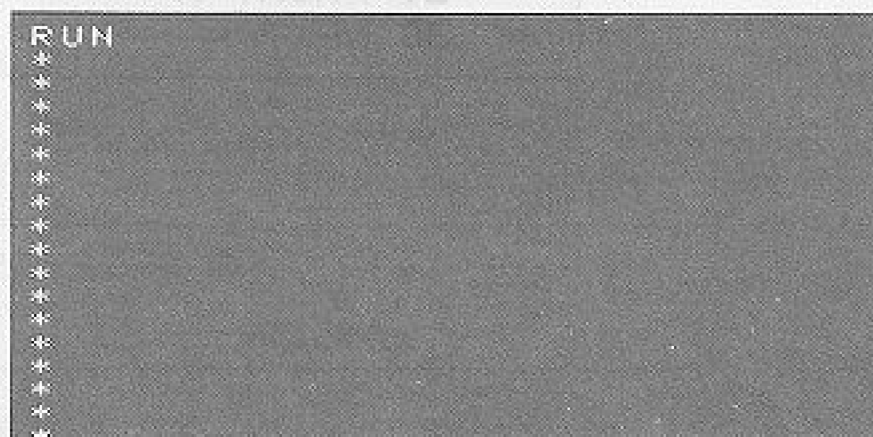
いくらでも考えられますね。ただし，ここでもあつかえる数は，－32767から32767までの整数です。

6－4 PRNT 命令，いろいろ

いままで出てきた PRINT 命令は，画面の左側はじから始まる表示ばかりでした。たとえば，次のようなプログラムを作ってください。NEW 命令でいままで入っていたプログラムを消してから，書き込んでください。

```
10 CLS
20 PRINT "*"←―― アスタリクス
30 GOTO 10
```

これは，GOTO 命令を使った無限ループですね。実行してみましょう。



画面の左はしだけに「*」(アスタリクス)が表示されています。プログラムの実行を止めてから次に進んでください。SHIFT RESET です。

●画面いっぱいに表示させたい。

こんどは「*」を画面いっぱいに表示させてみましょう。まずリストを取ってください。行番号20の最後に次のように「；」(セミコロン)を付けます。

```
10 CLS
20 PRINT "*";
30 GOTO 20   └―― セミコロンを付けます。
```

実行してみましょう。画面は，＊マークでいっぱいになりました。この；(セミコロン)は，セミコロンを付けられたもの(ここでは，「*」)を，右よこに続けて表示する場合に付けます。

```
RUN
********************************
********************************
********************************
********************************
********************************
********************************
********************************
********************************
********************************
********************************
********************************
********************************
********************************
********************************
********************************
********************************
********************************
```

● 8字分あけて表示させる

こんどは，8字分間をあけて「*」を表示させます。行番号20の；(セミコロン)を，(カンマ) に変えます。

```
20 PRINT "*",  ←―― カンマをつけます
```

実行してみましょう。

```
RUN
*       *       *       *
*       *       *       *
*       *       *       *
*       *       *       *
*       *       *       *
*       *       *       *
*       *       *       *
*       *       *       *
*       *       *       *
*       *       *       *
*       *       *       *
*       *       *       *
*       *       *       *
*       *       *       *
*       *       *       *
*       *       *       *
*       *       *       *
```

いかがですか，このように，(カンマ)を付けると，8文字分の間をあけて表示することができます。

6－5 もうひとつのカウンター

6－3では，［FOR～TO～NEXT］ 命令のループでカウンターを作りました。
こんどは，違う方法でカウンターを作ってみましょう。［NEW］ 命令を実行し，次のように，書きこんでください。

```
         10 LET CLS
      ┌→20 LET A=A+1
ループです│ 30 PRINT A,
      └→40 GOTO 20
```

実行してみましょう。画面は，行番号30に・(カンマ)が付いているので，8字分あきますがちゃんとカウントしています。

さて，カウンターはどれでしょう。もう分かりますね。行番号20と40で作られたループがカウンターです。
行番号20はこうです。

```
20 A=A+1
```

数学的にいうと，こんな式は成りたちませんね。でもプログラム文では，これで正しいのです。
これは，「Aに1をたした数を，Aに代入する。」ということですが，なんだかよく分かりませんね。

プログラムに新しく変数をセットしたとき（この場合は，変数Aです），プログラムを実行すると，その変数の値は自動的に0になります。では，わかりやすいように順を追って説明してみましょう。

①実行したとき A＝0。
②行番号20にきたとき，まず「A＝A＋1」の「A＋1」が計算されます。「0＋1」で1ですね。この1がAに代入されますから，　ここでA＝1となります。

③行番号30でAの値を表示し，行番号40のGOTO命令で，実行手順が行番号20へ戻ります。

④行番号20へ戻ったときは，A＝1ですね。そこで②と同じよう「A＋1」が計算されます。「1＋1」で2になりました。この 2 が新たにAに代入されて，A＝2になります。

このように，このプログラムは無限ループですから，無限に（といっても，あつかえる数は－32728～32767までですが）カウントして行くわけです。

6－6 IF～THEN で数を限ろう

● 100まで数えさせる。

6－5でやったカウンターを100まで数えさせた所で止めるようなプログラムを作ってみましょう。

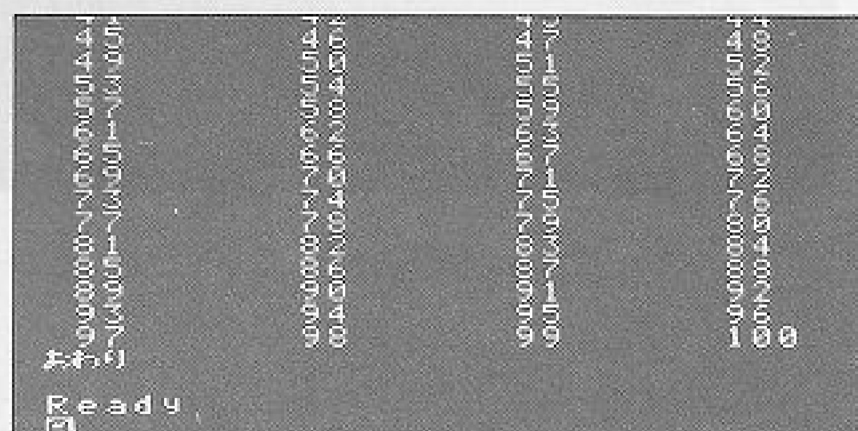
FOR～TO～NEXT 命令でのやり方は6－3で説明しました。こんどは，IF(イフ)～THEN(ゼン)命令を使って書いてみましょう。

それでは，先ほどの6－5のプログラムのリストを取ってください。消してしまった人はもう一度同じプログラムを書き込んでください。

リストがとれたら，次のプログラム文を追加させます。

```
35 IF A=100 THEN GOTO 50
50 PRINT "おわり"
60 END
```

まず，実行してみましょう。



ちゃんと，100までカウントしてプログラムの実行は，止まりました。これは，行番号35が実行されたためです。

```
35 IF A=100   THEN   GOTO 50
   もしA=100  なら   行番号50へ行け
```

この IF～THEN 命令は，条件判断をする命令で，「もしA＝100なら行番号50へ行きなさい」という意味になります。つまり，変数Aが100になるまでは，次の行のプログラムを実行し，変数Aが100になると行番号50へジャンプしますので画面に“おわり”と表示してプログラムの実行を終了します。

```
10 CLS
20 A=A+1
30 PRINT A,
35 IF A=100 THEN GOTO 50
40 GOTO 20
50 PRINT "おわり"
60 END
```

無限ループ

Aの値をみている。

6―7 条件判断・いろいろ

条件判断って何でしょう。あなたは，「明日晴れたらドライブへ行こうよ」なんてことを，言ったことはありませんか。
これをくわしく書くと，「もし，明日晴れたらドライブへ行くけれど，雨だったらどこへも行かない」と，なりますね。
条件判断とは，こんなものだと思ってください。これを， IF～THEN 命令で書くとこうなります。

IF（明日の天気）＝（晴れ） THEN（ドライブへ行く） ELSE（どこも行かない）

つまり，IFの後に書いた条件が成りたてば，THENの後に書いたことを実行し，条件が成りたたなければ，ELSEの後に書いてあることを実行します。
ELSE（エルス）というものが出てきましたが，これは， IF～THEN 命令と必ずいっしょに使われ，上に書いたようにIFの後に書いた条件が成りたたなかったときに，ELSEの後に書いたことを実行させます。ただし，ELSEは，必要がなければ省略してもかまいません。

● IF～THEN 命令の実際

IF～THEN 命令には，数の大小を比較する記号がよく使われます。分りやすいように，AとBという変数を使って説明しましょう。

記　号	書き方	意　　味
＝	A ＝ B	AとBは等しい
＞	A ＞ B	Aの方がBより大きい（Bの方がAより小さい）
＜	A ＜ B	Bの方がAより大きい（Aの方がBより小さい）
≧	A ≧ B	Aの方がBより大きいか，または等しい
≦	A ≦ B	Bの方がAより大きいか，または等しい
＜＞	A＜＞B	AとBは等しくない

この記号は，これから必ず使うものですから，覚えておいてください。これらの記号で比較される式を関係式といいます。

次に **IF～THEN** 命令の実際例をいくつか示します。参考にしてください。

○もし，AがBより大きければ，行番号30へ行き，それ以外だったら，行番号60へ行きなさい。

IF　A＞B　THEN　GOTO 30　ELSE GOTO 60

○もし，AとBが等しければ，「ひとしい」と表示し，それ以外だったら「ちがう」と表示しなさい。

IF　A＞B　THEN PRINT "ひとしい"　ELSE PRINT "ちがう"

○もし，Aが100だったら，「おわり」と表示し，それ以外だったら，行番号30へ行きなさい。

IF　A＝100　THEN PRINT "おわり"　ELSE GOTO 30

いろいろな条件判断ができますね。THENと，ELSEの後には，どんな命令文を書いてもかまいません。

こんなこともできます。

IF　A＝B　THEN IF　C＞D　THEN GOTO 100

これは，「もしAとBが等しければ，CとDの大小を比較して，もしCがDより大きければ，行番号100へ行きなさい。」ということです。しかし，あまり複雑なIF～THEN を書いたり，長い命令文を作るとプログラムが分りづらくなる恐れがあります。

Step 7

サイコロゲーム

RANDOMIZE　DIM

GOSUB　RETURN

READ　DATA

RND

7－1　ゲームのプログラム

いままでのStepは，BASIC-Ⅰになれてもらうために書きました。でもこれからは，違います。ゲームをしながら，そのゲームのプログラムを説明をしていきます。
始めは，〝サイコロゲーム〟を作ります。この〝サイコロゲーム〟は，ゲームとしては簡単なものですが，これからもっとむずかしいプログラムを作るために必要なことが，たくさん含まれています。

また，ここからは少しむずかしい言葉や説明も出てきますが，わかりにくければとにかく説明に出てくるプログラムを書きこんで実行してみてください。

7－2 RND(ランダム)は，ゲームの主役

サイコロゲームといっても，初めから画面にサイコロのグラフィックを作るのは大変ですね。そこで，サイコロの目を数字で表示するゲームを作ってみましょう。

●RND関数って？

サイコロというのは，1～6までの数がデタラメに出ますね。このデタラメな数を発生させる関数が，乱数関数です。乱数関数はRND(ランダム)関数ともいいます。

(※BASIC-I は，このRND関数以外に22種類の関数が使えます。*101* ページに関数一覧表がのっていますので，参考にしてください。)

このRND関数を使って，サイコロの目を出すプログラムは，次のようになります。

```
10 CLS
20 SAI=RND(5)+1  ←── サイコロ
30 PRINT SAI      └── サイコロには“0”の目はないので1から5の数に1をたしている。
40 GOTO 20
```

このプログラムは無限ループになっています。実行させると1から6までのいずれかの数が次々に表示されます。適当なところで SHIFT RESET でプログラムの実行を止めてください。

このように，RND 関数を使うと，希望する数の範囲で不規則な数を表示させることができます。たとえば，このプログラムで行番号20のRNDのあとの(　)内の数字を100とか，1000にすればその範囲内で不規則な数が表示できます。

7－3 "こんぴゅーた" とサイコロゲーム

サイコロを振らすことができたので，こんどは，コンピュータとサイコロゲームをしてみましょう。出た目の数の大きい方が勝ちです。
こんどは，次のようなプログラムになります。

```
                    10 CLS
                    100 REM わたしの さいの め
                    110 RANDOMIZE
自分用のサイコロ ──→120 LET SAI1=RND(5)+1
                    130 PRINT "わたし";SAI1
                    200 REM こんぴゅーたの さいのめ
                    210 RANDOMIZE
コンピュータ用の ──→220 LET SAI2=RND(5)+1
サイコロ
                    230 PRINT "こんぴゅーた";SAI2
```

実行してみましょう。画面には，あなたの振ったサイコロの目とコンピュータの振ったサイコロの目が表示されます。

```
わたし 4
こんぴゅーた 5

Ready
```

この数は乱数で発生していますのでこの通りになるとは限りません。

●審判が必要です。

コンピュータと自分用のサイコロは振れました。どっちが勝ったかは画面を見れば分りますが，その判断をコンピュータ自身にさせましょう。次のプログラム文を追加してください。

```
          300 REM かち まけ の はんだん
          310 LET M$="こんぴゅーたの かち"
審判です ┌320 IF SAI1=SAI2 THEN LET M$="あいこ"
         └330 IF SAI1>SAI2 THEN LET M$="わたしの かち"
          340 PRINT M$
```

審判は，前に出てきた IF～THEN 命令です。行番号310～330までの働きは次の様になります。

行番号 310……まずM＄に〝こんぴゅーたのかち〟と代入しておきます。
行番号 320……ＳＡ１とＳＡ２が等しければ，Ｍ＄に〝あいこ〟と代入します。ここで，Ｍ＄は，〝こんぴゅーたのかち〟から〝あいこ〟へ変りました。
また，ＳＡ１とＳＡ２が等しくなければ，次へ進みます。このときのＭ＄は，〝こんぴゅーたのかち〟のままです。
行番号 330……ＳＡ１の方がＳＡ２より大きければ，Ｍ＄に〝わたしのかち〟と代入します。それ以外は，次へ進みます。

前置きが長くなりました。実行してみましょう。ちゃんと勝ち負けをコンピュータが判断して表示してくれます。

```
わたし 5
こんぴゅーた 3
わたし の かち

Ready
```

文字にも変数があります。

行番号 310 に「Ｍ＄＝〝こんぴゅーたのかち〟」というプログラム文がありますが，これも変数の指定です。文字や記号も変数に代入できます。文字や記号を，" （ダブルコーテーションマーク）で囲み，変数の後ろに＄(ドルマーク)を付ければいいのです。
ドルマークを付けた変数は，文字をあつかうということで文字変数といいます。文字変数も数値変数と同じように，代入する文字を自由に変えることができます。また，変数名として使える文字も，先頭にアルファベット，ひらがな，グラフィックパターンをもってくれば，16字分組み合せた変数名が使えます。
ただし，代入できる文字の数には限りがあり，ダブルコーテーションマークで囲む文字数は，18字までです。

7－4 何度も使うプログラムは1つにまとめよう

さっき作ったプログラムには，自分用のサイコロとコンピュータ用のサイコロの2つのサイコロがありました。つまり，おたがいに1つずつのサイコロを持っていて，それぞれに振っていた訳です。
こんどは，サイコロを1つだけにして，あなたが振ってからコンピュータに振らせるようにしてみましょう。そうすれば，2つ必要だったサイコロが1つだけになるので，無駄がはぶけます。

●サブルーチンを使う。
1つのサイコロを2人で振るようなプログラムは，サブルーチンというものを使えば簡単に作ることができます。
まず，次のプログラム文を追加してください。ただし，行番号110と210は書き直します。

```
               110 GOSUB 400
               120 PRINT "わたし";SAI
               130 LET SAI1=SAI
               210 GOSUB 400
               220 PRINT "コンピュータ";SAI
               230 LET SAI2=SAI
               370 END
               400 REM さいころの さぶるーちん
               410 RANDOMIZE
サイコロ ──→ 420 LET SAI=RND(5)+1
               430 RETURN
```

いま追加したプログラムに，［GOSUB］（ゴーサブ）命令と［RETURN］（リターン）命令が新しくでてきました。
［GOSUB］命令と［RETURN］命令は，必ずペアーになって使われます。それは，プログラムの流れを，［GOSUB］命令で指定したサブルーチンへ飛ばしてから，［RETURN］命令で［GOSUB］命令の書いてある行番号の次の行番号へ戻すためです。

●サブルーチン

```
10 CLS
100 REM わたしの さいの め
110 GOSUB 400
120 PRINT "わたし";SAI
130 LET SAI1=SAI
200 REM こんぴゅーたの さいの め
```

```
210 GOSUB 400
220 PRINT "こんぴゅーた";SAI
230 LET SAI2=SAI
300 REM かち まけ の はんだん
310 LET M$="こんぴゅーたの かち"
320 IF SAI1=SAI2 THEN LET M$="あいこ"
330 IF SAI1>SAI2 THEN LET M$="わたしの かち"
340 PRINT M$
370 END
400 REM さいころの さぶるーちん
410 RANDOMIZE
420 LET SAI=RND(5)+1
430 RETURN
```

このプログラムの流れは次のようになります。

①行番号10でクリア・スクリーンされます。
②行番号110で行番号400のサブルーチンへ行きます。
③行番号400のサブルーチンでサイコロが1回振られて，行番号430の RETURN 命令で，行番号120へ戻ります。
④行番号130で，いま振られたサイコロの目の数をSAI 1に代入します。
⑤行番号210でまた行番号400のサブルーチンへ行きます。
⑥行番号400のサブルーチンでサイコロが振られ，行番号430の RETURN 命令で，行番号220へ戻ります。
⑦行番号230で，いま振られたサイコロの目の数をSAI 2へ代入します。
⑧後は，さっき説明した通り，勝ち負けの判断をしてから，結果を表示するプログラムです。

サブルーチンの働きは，こんなものです。うまく使えば，大変便利なものです。
※サブルーチンは，1つのプログラムの中に何個でも使われますし，サブルーチンの中にサブルーチンを使うこともできます。

これで，サイコロのプログラムは一応完成です。
せっかく作ったプログラムです。 SAVE 命令でカセット・テープにセーブしておきましょう。

7－5 どの目が何回でたか知りたい

RND 関数を使ったサイコロの作り方が分りました。こんどは，このサイコロを50回振らせてみて，何の目が何回出たかを表示してみましょう。
次のようなプログラムを書きこんでください。

```
10 LET CLS
20 FOR I=1 TO 50
30 LET SAI=RND(5)+1
40 IF SAI=1 THEN LET S1=S1+1
50 IF SAI=2 THEN LET S2=S2+1
60 IF SAI=3 THEN LET S3=S3+1
70 IF SAI=4 THEN LET S4=S4+1
80 IF SAI=5 THEN LET S5=S5+1
90 IF SAI=6 THEN LET S6=S6+1
100 NEXT I
110 PRINT "いちの めのかず は";S1
120 PRINT "に の めのかず は";S2
130 PRINT "さんの めのかず は";S3
140 PRINT "よんの めのかず は";S4
150 PRINT "ご の めのかず は";S5
160 PRINT "ろくの めのかず は";S6
```

実行してみましょう。画面は次のようになります。

```
いちの めのかず は 7
に の めのかず は 3
さんの めのかず は 10
よんの めのかず は 9
ご の めのかず は 5
ろくの めのかず は 12

Ready
■
```

このプログラムの，行番号40～90は条件判断で，１～６までのどの目が出たかを判断して何の目が何回出たか数えています。
たとえば，出た目が３だとすると，行番号60のＳ３＝Ｓ３＋１というカウンターで数えます。
一度実行してみてください。
でも，このプログラムは，同じ様なプログラム文がずいぶんありますね。そこで，いまのプログラムを もっと整理してみましょう。

7－6 配列変数を使おう

7－3のプログラムを整理すると，次の様なプログラムになります。

```
2 REM さいの め
5 CLS
10 DIM A$(6),B(6)
20 FOR I=1 TO 50
30 LET SAI=RND(5)+1
40 LET B(SAI)=B(SAI)+1
50 NEXT I
60 FOR I=1 TO 6
70 READ A$(I)
80 PRINT A$(I);"の めのかず は";B(I)
90 NEXT I
100 DATA "いち","に ","さん","よん","ご ","ろく"
```

（"に " の後の空白：スペース）

前のプログラムよりも，ずっとスッキリしました。
さて，プログラムを書きこんで実行してみましょう。
ちゃんと，前のプログラムと同じ働きをしていますね。

```
いちの めのかず は 7
に の めのかず は 8
さんの めのかず は 10
よんの めのかず は 8
ご の めのかず は 5
ろくの めのかず は 12

Ready
```

●たくさんのデータが，いっぺんにあつかえる「配列」

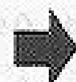
たとえば，同じ種類(文字あるいは数字)のデータがたくさんあり，それぞれを変数に代入したい。こんなとき，変数名をつけるとすると，データの数分変数名が必要です。これでは，あとで「このデータには，何って変数名をつけたっけ？」なんてことになりかねませんし，変数名をつけるためにプログラム文はデータの数だけ必要になってしまいます。

ここに学校の試験の成績の何回分かが，表になっているとします。これをあなたが見るときには，国語の1回目，国語の2回目，英語の4回目……などと考えながら見ると思います。答案用紙がごちゃごちゃになっているのではこうはいきませんね。

つまり，配列とは，試験の成績表の点数を見る要領と全く同じなのです。国語や英語などの科目を変数名A，B，Cとおきかえると，国語の1回目は，「Aの1」，国語の2回目は「Aの2」という呼びかたができますね。

(0)から使えますが使わなくてもかまいません。

	1回目	2回目	3回目
国語	75	80	90
英語	90	95	75
理科	80	85	75
社会	95	90	85



	1 (1)	2 (2)	3 (3)
A	75	80	90
B	90	95	75
C	80	85	75
D	85	90	85

BASICのプログラムの中では，このようにデータを並べることを配列といいます。そして今の表の各科目名が「配列変数」と呼ばれます。ただし，この配列変数をプログラムの中で扱うときにはあらかじめ，「ここでは配列を使います。」と宣言しなければいけません。これを「配列宣言」といって，DIM（ディメンジョン)命令で行います。
では，先ほどの２番目のプログラムを見てみましょう。行番号10がそうですね。

```
10 DIM A$(6),B(6)
```

扱うデータの数。「添字」といいます。ここにも変数がつかえます。このときは「添字変数」といいます。

命令

文字の配列表数

数字の配列表数

このプログラム文の意味は，「このプログラムでは配列を使います。１つはＡという配列変数で文字を扱います。扱うデータの数は０から６までの７つです。もう１つは，Ｂという配列変数で数字を扱います。扱うデータの数は０から６までの７つです。」ということになります。

さて，このようにして配列を宣言したプログラムで，実際に配列変数を用い，データを取り扱うときには，先ほどの試験の成績表で考えてみますと，国語の１回目は「Ａの１」と表わしましたが，これをA(1)，英語の３回目はB(3)，などと表わします。
いかがですか，チョットむずかしいかもしれませんね。

●配列変数の使い方

では，プログラムの行番号40を見てください。

```
40 LET B(SAI)=B(SAI)+1
```

これは，行番号10と50の FOR～TO～NEXT 命令のループに入っています。そして行番号30ではループが１まわりするたびに，サイコロを１回づつ振って，FOR～TO～NEXT 命令が終わるまで出た目の数を変数 SAI に50回代入し続けます。

たとえば，出た目の数が6だとすると，SAI = 6だから，B(6)= B(6)+ 1 となります。

B(0)～B(6)は，初めは0ですから，これで，B(6)= 1になりました。このようにサイコロを50回振って出た目の数をそれぞれ，Bに割りあてると，次のようになります。サイコロに0の目はないのでB(0)には何も割りあてません。

配列変数 B（SAI）	B(1)	B(2)	B(3)	B(4)	B(5)	B(6)
出た目の数	1	2	3	4	5	6

7−7 データをプログラムに書きこむ

変数に代入する数や文字は，「データ」といいました。何回も使うようなデータは，プログラム中に書きこんでそのつど変数に代入する方が便利です。

● READ (リード)命令と DATA (データ)命令

プログラム中に書きこんだデータを変数に代入するときは， READ 命令を使います。この命令は，「データを読む」という意味になります。そして，データを書いておくのには， DATA 命令を使い次のような書き方をします。

変数A，B，C＄，D，E＄ にそれぞれデータを読みこむ

```
10  READ   A, B, C$, D, E$
20  DATA   55, 32, "ABC", 63, "CDS"
```

文字変数は，ダブルコーテーションマークで囲み，データとデータの区切りに・(カンマ)を入れます。

7−7のプログラムの行番号60〜100を思い出してください。次のようでした。

```
60 FOR I=1 TO 6
70 READ A$(I)
80 PRINT A$(I);"の めのかず は";B(I)
90 NEXT I
100 DATA"いち","に ","さん","よん","ご","ろく"
```

（"に " の空白部分 ← スペース）

このプログラムは，FOR〜TO〜NEXT 命令のループの中の READ 命令で，配列変数のA＄にデータを読みこんでいます。A＄に読みこまれたデータを前にやったように，A＄(0)〜A＄(6)まで割りあてた表を作ると，次のようになります。

A＄	A(1)	A(2)	A(3)	A(4)	A(5)	A(6)
データ	いち	に	さん	よん	ご	ろく

Step 8

文字であそぼう

LEFT $　　RIGHT $　　MID $

CURSOR　　TAB

8－1 文字がばらばらに

こんどのプログラムは，18字までの文字を入力させて，１字づつ画面に表示させるものです。

まず次の様なプログラムを入力してみましょう。

```
10 REM もじ゛あそび゛
20 CLS
30 INPUT "18し゛まて゛のもし゛を いれる";MOJ$
40 LET KAZ=LEN(MOJ$)
50 FOR H=1 TO KAZ
60 LET HOJ$=LEFT$(MOJ$,H)
70 PRINT CURSOR(10,10);HOJ$
80 FOR C=1 TO 1000:NEXT C
90 NEXT H
100 GOTO 20
```

変数表

MOJ$……入力する文字列
HOJ$……表示される文字
KAZ………入力された文字数
H……………ループ
C……………カウンター

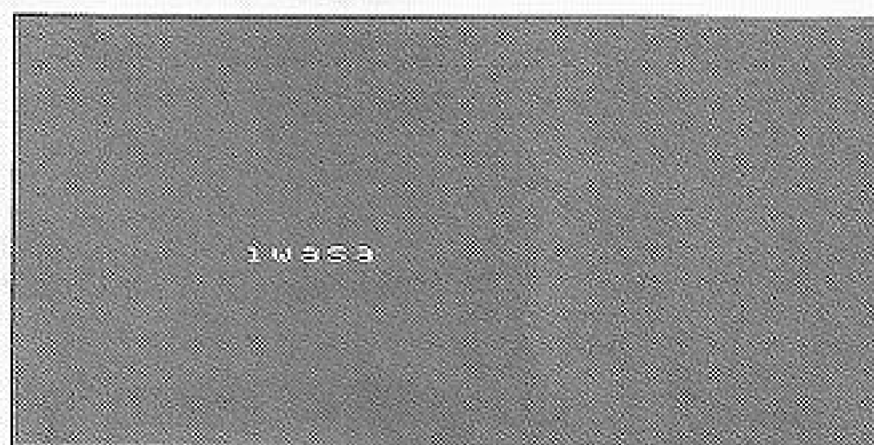
実行して，画面にどう表示されるか確めてみましょう。

18し゛まて゛のもし゛を いれる?

メッセージが出ました。あなたの名前を入れてみましょう。

18し゛まて゛のもし゛を いれる? iwasaki

RETURN を押すと，画面にいま入れた文字が最初から（左側から）順に表示されていきます。



全部表示されると，クリア・スクリーンされて入力まちになります。

●文字の数を数える。

```
40 LET KAZ=LEN(MOJ$)
```

行番号40に LEN （レングス）という文字が使われていますね。これは， LEN の後のカッコで囲まれた文字列の文字数を数える関数で「文字関数」といいます。文字が2つ以上つながっているものを文字列といいます。
さっき入力した文字列は，IWASAKI ですから，7文字ですね。したがって，KAZ＝7になります。

●文字列から文字を抜き取る。
行番号60を見てください。

```
60 LET HOJ$=LEFT$(MOJ$,H)
```

こんどは行番号60に LEFT＄（レフト）という文字が使われています。これは，文字列の左側から，好きなだけの長さの文字を抜き出す関数で，やはり「文字関数」です。この場合，ＭＯＪ＄＝IWASAKI ですから，IWASAKIという文字から変数「Ｈ」の値の数だけ，文字を抜き出してＨＯＪ＄に代入します。

変数「Ｈ」は，行番号40と90の FOR～TO～NEXT 命令にも使われています。Step 6－3 を思い出してください。

そこで「H」の値は1からKAZの値まで1つづつ増えますから，HOJ$には，IWASAKIという文字列の左側から，次のように代入されてきます。

1回目のループ　　HOJ$=I
2回目のループ　　HOJ$=IW
3回目のループ　　HOJ$=IWA
　　⋮　　　　　　　⋮
　　⋮　　　　　　　⋮
7回目のループ　　HOJ$=IWASAKI

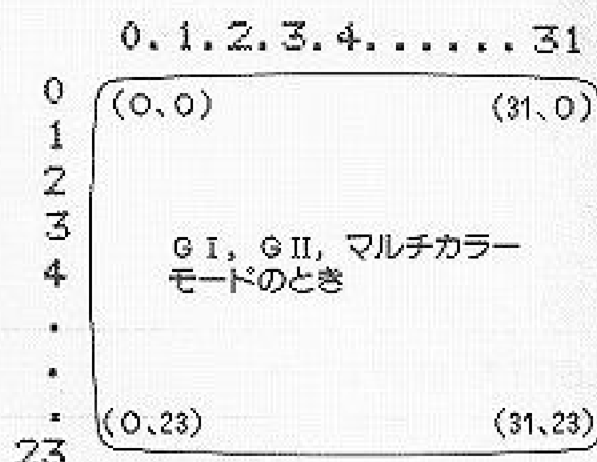
これを行番号70の [PRINT] 命令で表示させているのです。でも今までとは表示のされかたが違っていますね。これは，[PRINT] 命令の後に CURSOR 関数 というものが入っているからです。

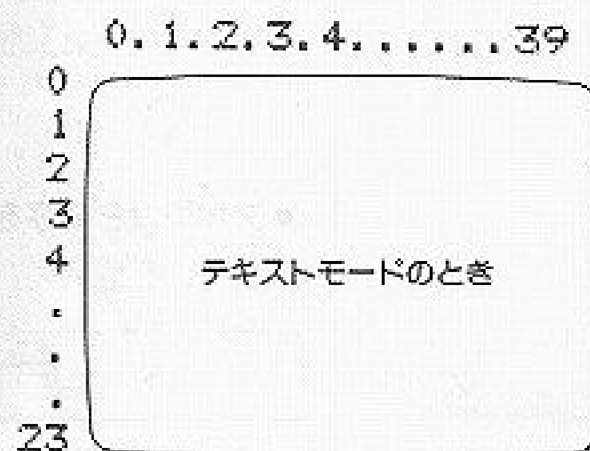
●画面上の好きな位置に表示させる。

CURSOR（カーソル）とは，その後のカッコで囲まれた位置に表示させる関数で，[PRINT] 命令と組み合わせて使います。

たとえば，次の様なプログラム文を実行すると "a" は画面の中央に表示されます。

```
print cursor(16,12);"a"
```





いま，あなたが見ている画面はこのコンピュータでは，GIモードといって通常使われる画面です。このときの画面は，よこ（0～31番），たて（0～23番）までの番号が付けられています。

CURSOR 関数は，よことたてを，画面上の番号で指定し，文字などを表示させる位置を決める命令です。たとえば，aを画面の右上に表示させるときは，

```
print cursor(31,0);"a"
```

となります。

つまり，位置を指定するときは，「CURSOR(よこの番号，たての番号)」と書きます。
これは，むずかしくいうと，座標指定と呼ばれるもので，よこをX，たてをYであらわし CURSOR(X，Y)と書きます。
注意してほしいのは，Xを32以上，Yを24以上指定すると，表示したいものが画面の外に出てしまい，見えなくなってしまいます。

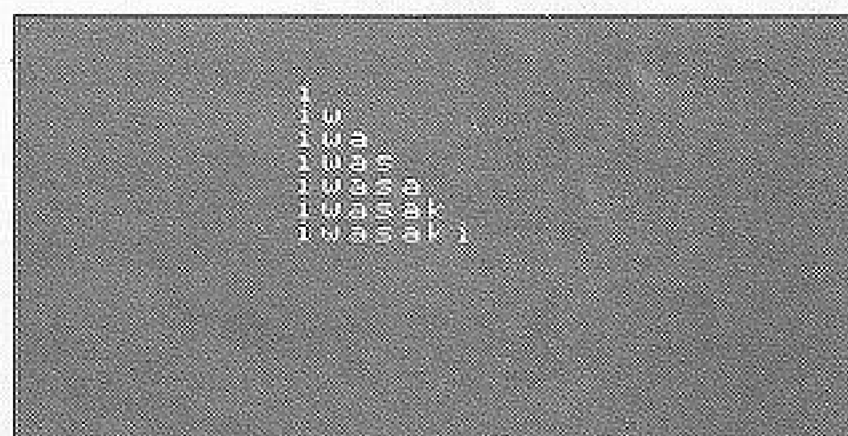
もうひとつの座標指定・TAB

CURSORは，よこ(X)とたて(Y)を使って画面上の一点を指定できましたね。このほかに，画面上のよこ(X)の位置だけを指定する方法があります。

画面のよこの位置だけを指定するときは， TAB （タブ)を使います。
8−1のプログラムを TAB を使って書き直すと，行番号70は，次のようになります。

```
70 PRINT TAB(10);HOJ$
```

このプログラムを実行させると画面は，右の様になります。



CURSORで座標を指定すると画面はスクロールしませんが，TABで指定すると，スクロールされてしまい，画面上の一点に止ったまま表示させることはできません。

8－3 いろいろな文字関数

さて，このプログラムの中にでてきた LEN ， LEFT $ を文字関数といいましたね。この文字関数とは，文字を専門に扱う関数のことで，他に RIGHT $ や MID $ などがあります。

●文字列の右側から抜き取る。

それでは，今のプログラムを RIGHT $ 関数を使って書きかえてみましょう。使い方は LEFT $ と同じです。行番号60は次の様になります。

```
60 LET HOJ$=RIGHT$(MOJ$,H)
```

さて，プログラムを実行するとどうなるのでしょう。自分で確認して下さい。

●文字列のまん中から抜き取る。

これは，MID $（ミドル）関数を使います。これは，LEFT $ ， RIGHT $ と使い方が，すこし違います。

たとえば，次のような文字数10の文字列があったとします。

文字列	A	B	C	D	E	F	G	H	I	J
文字数(左から数える)	1	2	3	4	5	6	7	8	9	10

これのCからGまでを抜き取るときは，左から数えて３番目から７番目を抜き取ると指定します。

MID $ ("ABCDEFGHIJ", 3, 7)

最初から最後まで抜き取るときは

MID $ ("ABCDEFGHIJ", 1, 10)

となります。

さっきのプログラムに使ってみましょう。行番号60は次のようになります。

```
60 LET HOJ$=MID$(MOJ$,1,7)
```

これも，実行して，確かめてみてください。

Step 9

サイコロのグラフィック

INKEY $

STCHR

$（ラベル）

9—1 サイコロの形を作ろう

さて，こんどはグラフィック・パターンを使って，Step 7 で説明した〝サイコロゲーム〟を本格的なゲームに作りかえてみましょう。

Step 7 でやったサイコロのプログラムをロードしてください。セーブしていなかった人は，71 ページにリストが載っていますので，プログラムを書きこんでください。

●サイコロの形はグラフィック・パターンで作る。

次のプログラムリストの ▒ のところを追加してください。サイコロの形は，グラフィックモードのグラフィックパターンを使います。（ FUNC 3 ）

```
5 REM さいころ
10 CLS
100 REM わたしの さいの め
110 GOSUB 400
120 PRINT"わたし";SAI
130 LET SAI1=SAI
140 GOSUB 500
200 REM こんぴゅーたの さいの め
210 GOSUB 400
220 PRINT "こんぴゅーた";SAI
230 LET SAI2=SAI
240 GOSUB 500
300 REM かち まけ の はんだん
310 LET M$="こんぴゅーたの かち"
320 IF SAI1=SAI2 THEN LET M$="あいこ"
330 IF SAI1>SAI2 THEN LET M$="わたしの かち"
340 PRINT M$
350 IF INKEY$=" " THEN GOTO 10      ← キーの押されるのを待っている
360 GOTO 350
370 END
400 REM さいころの さぶるーちん
410 RANDOMIZE                       ← サイコロ
420 LET SAI=RND(5)+1                ← サイコロ
430 RETURN
500 REM さいころの ひょうじ
510 IF SAI=1 THEN GOSUB 1000
520 IF SAI=2 THEN GOSUB 1100
530 IF SAI=3 THEN GOSUB 1200
540 IF SAI=4 THEN GOSUB 1300
550 IF SAI=5 THEN GOSUB 1400
560 IF SAI=6 THEN GOSUB 1500
580 RETURN
```

ダブルコーテーションマーク

```
1000 PRINT "┌─────┐"
1010 PRINT "│     │"
1020 PRINT "│  ●  │"
1030 PRINT "│     │"
1040 PRINT "└─────┘"
1050 RETURN
1100 PRINT "┌─────┐"
1110 PRINT "│    ●│"
1120 PRINT "│     │"
1130 PRINT "│●    │"
1140 PRINT "└─────┘"
1150 RETURN
1200 PRINT "┌─────┐"
1210 PRINT "│    ●│"
1220 PRINT "│  ●  │"
1230 PRINT "│●    │"
1240 PRINT "└─────┘"
1250 RETURN
1300 PRINT "┌─────┐"
1310 PRINT "│●   ●│"
1320 PRINT "│     │"
1330 PRINT "│●   ●│"
1340 PRINT "└─────┘"
1350 RETURN
1400 PRINT "┌─────┐"
1410 PRINT "│●   ●│"
1420 PRINT "│  ●  │"
1430 PRINT "│●   ●│"
1440 PRINT "└─────┘"
1450 RETURN
1500 PRINT "┌─────┐"
1510 PRINT "│●   ●│"
1520 PRINT "│●   ●│"
1530 PRINT "│●   ●│"
1540 PRINT "└─────┘"
1550 RETURN
```

サイコロパターン
グラフィックパターンを使います

サイコロパターンのサブルーチン

ちょっと長いですが，がんばってください。それでは，このプログラムを少し説明してみましょう。

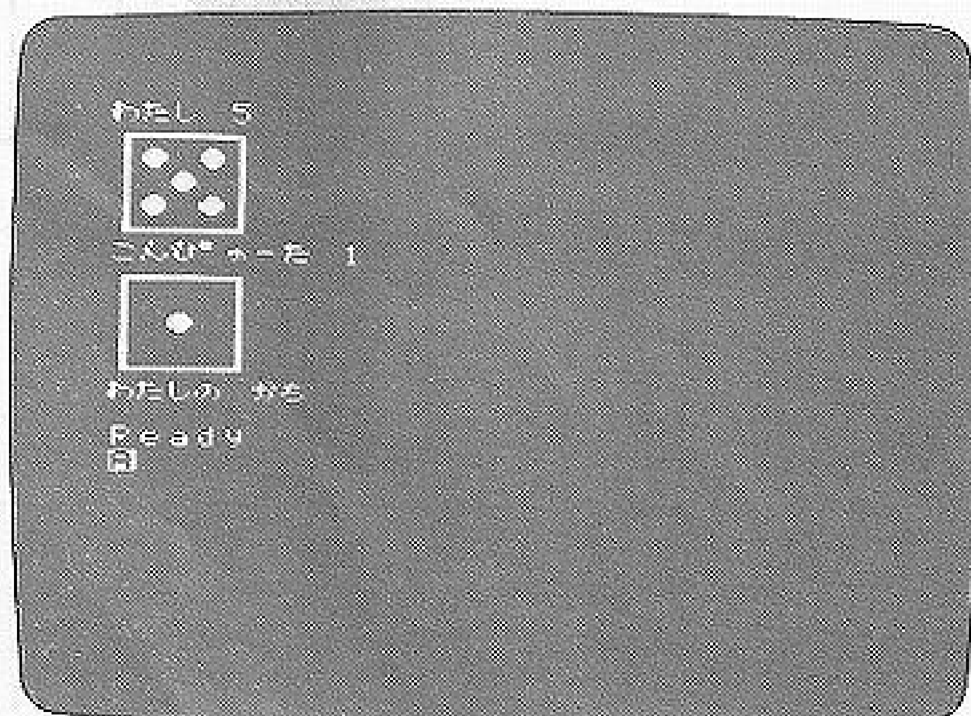
行番号 500～580 は，条件判断で，サイコロの目の数によって，どのサイコロパターンを表示させるか判断するプログラムです。これはサブルーチンになっています。
行番号1000～1550は，サイコロのパターンです。これは，行番号 500 ～ 580 のサブルーチンになっています。
ここで，一つだけ新しい関数が出てきました。行番号 350 の　**INKEY$**　です。

```
350 IF INKEY$=" " THEN GOTO 10
360 GOTO 350
```

（" " はスペースです）

これは，プログラム実行中に押されたキーを読みこむ関数で，この場合，SPACE キーが押されると行番号10へ行き，初めからプログラムを実行していきますが，SPACE キーが押されなければ，行番号360の GOTO 命令により，無限ループが作られていますので，いつまでも行番号350と360をくり返しているだけです。つまり，SPACE キーの押されるのを待ち続けるのです。
ここで，スペースの変りに，ダブルコーテーションマークで「A」という文字を囲めば，A のキーが押されるのを待つプログラムになります。

さて，前おきが長くなりましたが実行させてみましょう。サイコロが数字だけではなく，パターンで表示されましたね。もう一度やるときは，SPACE キーを押してください。



9―2 サイコロをクルクル回そう。

こんどは，サイコロに動きをつけてみましょう。今のプログラムに，次のプログラム文を追加してください。

```
                        ┌──────── スペース
150 IF INKEY$=" " THEN GOTO $HIT
160 PRINT "↑↑↑↑↑↑↑↑"←──── SHIFT  CTRL  ↑
170 GOTO 100
205 $HIT
```

行番号150は，さっきやった SPACE キーの入力待ちですが，行き先きの$ＨＩＴというのは，ＨＩＴという変数の先頭に$(ドルマーク)を付けたもので，文字変数とは違います。

これは，「ラベル」といって，GOTO 命令や，GOSUB 命令の後に書いて，その行き先にも同じラベル名を書いておけば，プログラムの実行手順を変えることができます。複雑なプログラムになった場合，ラベル名を GOTO 命令や GOSUB 命令で行く先のプログラムの内容に合ったものにしておけば，後でプログラムを見るときに，分りやすいプログラムを作ることができます。

ここでは，SPACE キーをＨＩＴ(ヒット)したら，コンピュータのサイコロを表示させたかったので，ラベル名を「$ ＨＩＴ」にしました。

行番号160は，スクロールを止める命令文です。これは，プログラム中で CTRL キーを使った例です。このようにプログラム中で CTRL キーを使う場合，PRINT 命令の後にダブルコーテーションマークで，使いたい制御コード(付録*103*ページを見てください。)を CTRL SHIFT といっしょに押せば，使うことができます。

行番号205は，行番号150の$ ＨＩＴの行き先です。

それでは実行して，追加したプログラムの働きを確かめてください。やめたいときは，SHIFT RESET を押します。

9－3 サイコロの目に色を付けよう

こんどは，サイコロの目に色を付けてみましょう。次のプログラム文を追加してください。

```
20 PRINT "▨"
30 FOR I=1 TO 6
40 STCHR "4040404040404040" TO &E1,I
50 NEXT I
```

行番号20は，さっきもでてきた制御コードで，画面をＧⅡモードにする命令文です。
行番号40のSTCHR（セットキャラクター）は，色を付ける命令で，書き方は次のようになります。

```
40 STCHR "4040404040404040" TO &E1,I
```

色の指定（"4040404040404040"）　色を付けるもの（&E1）　画面位置（I）

※色の指定…………付録のカラーコード表（ *96* ページ）を参考にしてください。
※色をつける物……付録のアスキーコード表（16進数）を参考にしてください。
※画面位置…………ＧⅡモードの画面は次のように上・中・下の３つに別れていて，それぞれで色を指定できます。

上	4
中	5
下	6

この場合，Ｉに４だけ代入すると画面の上の方では，青くなりますが，あと，中・下のところでは，色が付かなくなります。
このプログラムでは，行番号40～50のFOR～TO～NEXT命令でＩを４～５にして，画面全部でサイコロの目に色が付くようにしています。

おわりに

いかがでしたか，紙面の都合で十分な説明ができなかったところもありますが，BASIC Ⅰの概略がおわかりいただけたでしょうか。初めての方には少しむずかしかったかも知れませんが，説明に出ているプログラム例を，実行させた結果と比較して考えていただければ何となく理解していただけることと思います。あとは，とにかくいろんなプログラムにチャレンジしてみることです。まずチャレンジして，何となく理解できたら今度は応用してみる。これがBASICの上達方法です。がんばってください。
本書では，BASICを既にご存知の方のために，付録の頃に「命令語一覧表」，「関数一覧表」や，各種のコード表を載せてあります。また，BASIC Ⅰの高度なテクニックを使った「ＵＦＯゲーム」のプログラム例が載せてあります。興味のある方はぜひチャレンジしてみてください。

１９８２年１０月　筆者

付 録

UFOゲーム

飛んでくる隕石をよけながら、インベーダーを捕えてください。プログラムは、RUN で実行します。

遊び方

円盤を上に動かす……… ■

円盤を下に動かす……… ■

得点

インベーダーを1つ捕えるたびに10点ずつ上がります。約1200点以上になると、隕石の数が増えインベーダーを1つ捕えるたびに100点の得点になります。

注意

あまり長い間、インベーダーを捕えないとゲームが終って、「Ready」と表示されます。

```
5 REM ufo game
10 READ SC
20 OUT &20,&F6
30 LET X=5
40 LET A=PEEK(&701A)
50 POKE &701A,A AND &EF
60 PRINT "■■■";:LET SL=100
70 VIEW 0,5,31,23:LET STC=&40
100 LET UX=32:LET UY=96:LET LV=3
110 GOSUB 10000
120 FOR LP=0 TO 1
140 PRINT CURSOR(30,RND(18));"◆" LIST
141 IF RND(9)<4 THEN PRINT CURSOR(30,RND(18)+5);"△";
142 FOR I=1 TO LV
150 PRINT "■";
154 FOR J=1 TO 2
160 GOSUB 1000:GOSUB 2000
164 NEXT J:NEXT I
170 LET SL=SL-1
180 IF SL<0 THEN END
190 IF LP=0 THEN GOTO 220
200 STCHR "183c7edbdb7e8142" TO &7F,1
210 GOTO 300
220 STCHR "183c7edbdb7e4281" TO &7F,1
300 NEXT
400 GOTO 120
1000 LET S=PEEK(&702B)
1010 IF S=51 THEN GOSUB 1100
1030 IF S=46 THEN GOSUB 1200
1060 LOC 0 TO UX,UY
```

```
1070 RETURN
1100 IF UY>=42 THEN LET UY=UY-2
1110 RETURN
1200 IF UY<=182 THEN LET UY=UY+2
1210 RETURN
2000 LET UAD=UY/8*32+UX/8+&3800
2010 LET UAD=UAD+(UY AND 4)*8+(UX AND 4)/4
2016 LET BCHR=VPEEK(UAD)
2020 IF BCHR=225 THEN GOTO 3100
2030 IF BCHR=127 THEN GOTO 2200
2040 RETURN
2200 VIEW
2210 LET O=O+SC:LET SL=SL+10
2215 PRINT "▣"
2220 PRINT CURSOR(9,2);O;
2230 VPOKE UAD,32
2240 LET UX=UX+1
2250 IF UX<160 THEN GOTO 2400
2260 IF LV<=1 THEN GOTO 2400
2270 LET LV=LV-1
2280 LET UX=32
2290 READ SC
2300 LET STC=STC+&40
2310 VPOKE &3B9C,STC
2400 VIEW 0,5,31,23
2410 RETURN
3100 VIEW
3106 VPOKE UAD,32
3110 LOC 0 TO 256,0
3130 FOR BL=1 TO 3
3140 FOR BB=1 TO 2
3150 LOC BB TO UX,UY
3160 LOC 3-BB TO 256,0
3170 LOC BB TO UX,UY
3200 GOSUB 30000
3210 NEXT BB
3220 NEXT BL
3230 LOC 2 TO 256,0
3240 LET X=X-1
3260 PRINT CURSOR(10+X,3);"  ";
3280 IF X=0 THEN GOTO 4000
3300 GOTO 2240
4000 PRINT CURSOR(12,10);"Game Over"
4010 OUT &20,&FF
4020 END
```

```
10000 REM
10050 VIEW
10110 VPOKE &3B9C,STC
10130 STCHR "182442427effff66" TO 0
10136 STCHR "182442427effff66" TO &82,1
10140 STCHR "3c66e79999e7663c" TO 1
10150 STCHR "c399186666l899c3" TO 2
10160 STCHR "183c7edbdb7e4281" TO &7F,1
10200 FOR I=0 TO 2
10210 SCOD I,I
10220 SCOL I,2^(I+1)
10250 NEXT I
10260 PRINT CURSOR(3,2);"■Score:■";
10266 PRINT CURSOR(3,3);"Left :♠♠♠♠♠";
10270 PRINT "■┏";
10280 FOR I=1 TO 30
10290 PRINT "━";
10300 NEXT I
10310 PRINT "┓";
10320 FOR I=1 TO 3
10330 PRINT "┃"
10340 NEXT I
10350 PRINT "┗";
10360 FOR I=1 TO 30
10370 PRINT "━";
10380 NEXT I
10390 PRINT "┛";
10510 VIEW 0,5,31,23
11000 RETURN
30000 OUT &20,&9F:OUT &20,&BF
30010 OUT &20,&E7:OUT &20,&F0
30020 FOR FRQ=1 TO 20
30030 OUT &20,&C0
30040 OUT &20,FRQ
30045 FOR T=1 TO 60:NEXT
30050 NEXT FRQ
30060 OUT &20,&F6
30070 RETURN
31000 DATA 10,100,200
```

色コード表

色は次のコード表を見て指定してください。この指定方法は、GⅡモードで行ってください。

```
STCHR "80 80 80 80 80 80 80 80" TO &E1  4
      └───────────┬───────────┘    └─┬─┘
                  └ 色コード          └ キャラクターコード
```

たとえば、上のように指定すると、●（キャラクターコードで&E1）が赤になります。
白ぬきにするときは、

STCHR "08 08 08 08 08 08 08 08" TO&E1、4

とすると、◘のようになります。

色コード表

色コード番号	0	1	2	3	4	5	6	7
色	透明	黒	緑	淡い緑	濃い青	淡い青	濃い赤	シアン
色コード番号	8	9	A	B	C	D	E	F
色	赤	淡い赤	濃い黄	淡い黄	濃い緑	マゼンタ	グレー	白

キャラクターコード表

アスキーコード表と対応していますが，特にキャラクターの指定をするときに便利なものです。このコード表は16進で書いてありますので，使うときには，&マークを付けてください。

例．●の指定……&E1
　　Mの指定……&4D

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		[illegible]		0	@	P	`	p	▒	▁		ー	た	み	※	▬
1	[illegible]	[illegible]	!	1	A	Q	a	q	○	▂	。	あ	ち	む	●	▬
2	[illegible]	[illegible]	"	2	B	R	b	r	♠	▃	「	い	つ	め	♥	▤
3	[illegible]	[illegible]	#	3	C	S	c	s	♣	▄	」	う	て	も	♦	▤
4	[illegible]	[illegible]	$	4	D	T	d	t	～	▌	、	え	と	や	〈	▌
5	[illegible]	[illegible]	%	5	E	U	e	u	┴	▎	・	お	な	ゆ	／	▌
6	[illegible]	[illegible]	&	6	F	V	f	v	┬	▒	を	か	に	よ	＼	▐
7	[illegible]	[illegible]	'	7	G	W	g	w	│	▇	年	き	ぬ	ら	╳	▩
8	[illegible]	[illegible]	(	8	H	X	h	x	─	◆	円	く	ね	り	＝	▖
9	[illegible]	[illegible]	)	9	I	Y	i	y	┤	▼	日	け	の	る	┤	▗
A	[illegible]	[illegible]	*	:	J	Z	j	z	├	▲	時	こ	は	れ	┼	⧗
B	[illegible]	[illegible]	+	;	K	[	k	{	┼	▶	分	さ	ひ	ろ	╪	⋈
C	[illegible]	→	,	<	L	¥	l	¦	┌	╭	ゃ	し	ふ	わ	◢	◢
D	[illegible]	←	-	=	M	]	m	}	└	╰	ゅ	す	へ	ん	◥	◥
E	[illegible]	↑	.	>	N	^	n	~	┐	╮	ょ	せ	ほ	゛	◣	◣
F	[illegible]	↓	/	?	O	_	o	◄	┘	╯	っ	そ	ま	゜	◤	◤

命令語一覧表

ダイレクト命令

	命令語	書き方	使用例
1	AUTO	AUTO M,N	AUTO 10 .AUTO 100,10
2	CLEAR	CLEAR	CLEAR 256, &7E00
3	CLS	CLS	CLS
4	CONT	CONT	CONT
5	DEL	DEL M,N	DEL 10 .DEL 10,50
6	LIST	LIST M,N	LIST 10 .LIST 10,100 .LIST ,50
7	LIST #2,	LIST #2,M,N	LIST #2, LIST #2 10,100
8	NEW	NEW	NEW
9	RUN	RUN .RUN N	RUN .RUN 100

入出力命令

	命令語	書き方	使用例
1	CHAIN	CHAIN"ﾌｧｲﾙﾒｲ"	CHAIN"UFO"
2	DATA	DATA N,M.----	DATA 1,34,"CD"
3	INPUT	INPUT N,M,-- .INPUT N$,M$--	INPUT A,B .INPUT A,B
4	OLD	OLD "ﾌｧｲﾙﾒｲ"	OLD "UFO" .OLD"UFO"
5	OUT	OUT N,M	OUT &20,&3F
6	PRINT	PRINT N,M,-- .PRINT N$,M$,--	PRINT A,B .PRINT "UFO",33
7	PRINT #2,	PRINT #2,	PRINT #2,
8	READ	READ N,M----	READ A,B,C$
9	RESTORE	RESTORE N	RESTORE 100
10	SAVE	SAVE"ﾌｧｲﾙﾒｲ"	SAVE"UFO"
11	SAVE	SAVE"ﾌｧｲﾙﾒｲ",&M,&N,1	SAVE"UFO",&2000,&3FFF,1
12	TAPE	TAPE	TAPE
13	VERIFY	VERIFY"ﾌｧｲﾙﾒｲ"	VERIFY"UFO"

命令語や関数で、　　　がかかっているものは、本文中に説明があります。　　　がかかっていないVPOKE命令、POKE命令は、 マシン語の知識がないと使えません。間違った命令のし方をすると、壊れることがありますので注意してください。

	コメント
1	行番号の自動発生
2	全変数の初期化。BASICの作業域の指定。
3	クリア・スクリーン
4	プログラムの実行中　STOP命令で止めたプログラムを止まった行番号から再実行される。
5	プログラムの一部を消す。
6	プログラムリストを、画面に表示させる。
7	プログラムリストをプリンターに打ち出す。
8	全プログラムを消す。
9	プログラムの初めから実行する。行番号Nから実行する。

	コメント
1	テープにセーブしたプログラムを、実行中に呼び出しロードする。ただし、ロードした時点で変数は初期化されるため連続した計算などは行えない。
2	READ文で読みとるデータを書きこむ。
3	プログラムの実行中にデータをキーボードからのキー入力で読みこみ、変数に代入する。
4	テープにセーブされているプログラム、又はデータをロードする。
5	Nで指定されたI/Oポート（入出力装置）にMの数字で与えられた値を出力する。
6	画面に表示させる。
7	プリンタに出力させる。
8	DATA文に書かれたデータを、DATAに書いてある順に読みとる。
9	READ文で読むDATA文を指定する。RESTORE　100とすると行番号100に書かれたDATAから読みとる。
10	プログラムをテープにセーブする。
11	使用例のようにすると、画面に表示されているデータをテープにセーブする。（約1分ぐらいかかる）
12	アプリケーションプログラムの実行（本体の取扱い説明書参照）
13	正しくテープにセーブされているか確認する。テープに入っているプログラムのファイル名を表示させる。

プログラム命令語

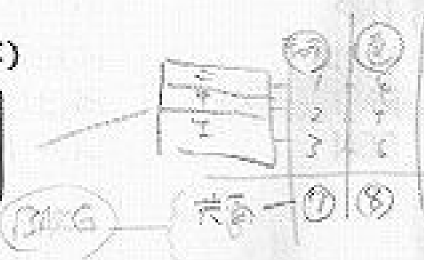
	命令語	書き方	使用例
1	CALL	CALL N	CALL
2	DIM	DIM N(M)　　.DIM N$(M)	DIM A(10)　　.DIM A$(10)
3	END	END	END
4	FOR-TO-STEP	FOR N=M TO A STEP B	FOR N=1 TO 100 STEP 2
5	GOSUB	GOSUB N　　.GOSUB $N	GOSUB 100　　.GOSUB $HIT
6	GOTO	GOTO N　　.GOTO $N	GOTO 100　　.GOTO $NIT
7	IF-THEN-ELSE	IF N=M THEN 〈メイレイ〉 ELSE 〈メイレイ〉	IF A=1 THEN GOTO 100 ELSE B=1
8	LET	LET N=シキ	LET N=N+1
9	NEXT	NEXT N	NEXT N
10	POKE	POKE N,M	POKE &2000,&3F
11	RANDOMIZE	RANDOMIZE	RANDOMIZE
12	REM	REM	REM UFO NO PRGRAM
13	RETURN	RETURN	RETURN
14	STOP	STOP	STOP

画面制御命令

	命令語	書き方	使用例
1	LOC	LOC N TO X,Y	LOC 0 TO 256,192
2	MAG	MAG N	MAG 3
3	SCOD	SCOD N,C	SCOD 0,255
4	SCOL	SCOL N,C	SCOL 0,4
5	STCHR	STCHR "　　" TO N,M	STCHR"4040404040404040"TO&1E,6
6	VIEW	VIEW X0,Y0,X1,Y1	VIEW 0,0,31,23
7	VPOKE	VPOKE N,M	VPOKE &3B80+C/8,4*16+0

	コメント
1	Nで与えられたアドレスから始まるプログラムを呼びだす。（BASICからマシン語を呼びだす。）
2	配列を宣言する。
3	プログラムの実行を止める。
4	ループを作る。 例）Nが100になるまで1から2つおきにループを実行する。
5	指定されたサブルーチンへ飛ぶ。
6	指定された行番号へ飛ぶ。
7	条件判断を行う。 例）もしAが1ならば、（A＝1）行番号100へ飛びなさい。
8	代入命令
9	FOR〜TO〜STEP文とペアーで使われ、FOR〜TO〜STEP文とNEXT文で囲まれたプログラムをループ処理する。
10	Nで与えられたアドレスに、Mの値を書きこむ。
11	乱数の初期値を変更する。
12	プログラムの実行とは関係なしに、コメントを入れる。
13	GOSUB文の書いてあった行番号の次の行番号へ戻る。
14	プログラムを一時止める。再実行は、CONTを使う。

	コメント
1	スプライトの画面上の位置を決める。 N＝0〜31（スプライト番号） X＝−32768〜32767（ドット単位） Y＝−32768〜32767（ドット単位） ただし、画面上で見えるのは、X＝0〜255　Y＝0〜191まで。
2	スプライトの大きさを変える。（N＝0〜3）
3	スプライトに図形コードを割あてる。 N＝0〜31　（スプライト番号） C＝0〜255（図形コード）
4	スプライトの色を指定する。 N＝0〜31（スプライト番号） C＝0〜15（カラーコード）
5	キャラクター・パターンを文字コードに割りあてる。色の指定をする（GIIモード） " "＝キャラクター・パターン又は、色の指定。 N　＝文字コード（キャラクター・コード表参照） M　＝1〜3（キャラクター・パターンの割りあて）。4〜6（色の割りあて）
6	画面を（X0、Y0）と(X1、Y1）で指定された、四角形で区切る。
7	Nで与えられたVRAMのアドレスに、Mの値を書きこむ。



関数一覧表

文字関数

	関 数	書き方	使用例
1	ASCII	ASCII(X$)	A=ASCII("A")
2	CHR$	CHR$(N)	A=CHR$(255)
3	HEX$	HEX$(N)	A$=HEX$(16)
4	INKEY$	INKEY$	A$=INKEY$
5	LEFT$	LEFT$(X$,N)	A$=LEFT$("ABCDEFG",7)
6	LEN	LEN(X$)	A=LEN(A$)
7	MID$	MID$(X$,N,M)	A$=MID$("ABCDEFG",1,7)
8	RIGHT$	RIGHT$(X$,N)	A$=RIGHT$("ABCDEFG",7)
9	VAL	VAL(X$)	A=VAL(A$)

オペレート関数

	関 数	書き方	使用例
1	CURSOR	CURSOR(X,Y)	PRINT CURSOR(16,12);A
2	ERR	ERR	PRINT ERR
3	ERRL	ERRL	PRINT ERRL
4	ERRL$	ERRL$	PRINT ERRL$
5	PEEK	PEEK(N)	A=PEEK(&7000)
6	TAB	TAB(X)	PRINT TAB(10);A
7	VPEEK	VPEEK(N)	A=VPEEK(&3800+Y*32+X)

数値関数

	関 数	書き方	使用例
1	ABS	ABS(N)	A=ABS(-5)
2	FRE	FRE(N)	A=FRE(0)
3	INP	INP(N)	A=INP(0)
4	NUM$	NUM$(X)	A$=NUM$(5)
5	RND	RND(N)	A=RND(10)
6	SGN	SGN(N)	A=SGN(3)
7	TIME	TIME	PRINT TIME

	コメント
1	カッコで囲んだ文字をコード番号に変える。（アスキーコード表参照）
2	カッコで囲んだコード番号を文字に変える。ASCIIと逆の働きをする。 例）ASCII（"a"）＝97　CHR$(97)＝a
3	カッコで囲んだ数値を16進数に変える。（10進、16進の対応は、アスキーコード表にのっています。）
4	プログラム実行中、キーボードから1文字分の入力を読みとる。
5	Xに書いた文字列を左側からN文字分与える。
6	カッコで囲んだ文字列の数を数える。
7	Xに書いた文字列をN番目の文字からM文字分与える。 例）MID（ABCDEF、2、3）左から2番目の文字から3文字分与える。
8	Xに書いた文字列を右側からN文字分与える。
9	カッコで囲んだ文字列の長さを与える。

	コメント
1	画面上の座標（X、Y）を指定する。 （X＝0～31（G1、G2、マルチカラー）、0～39（T） Y＝0～23）
2	エラーのコード番号を与える。
3	エラーの発生した行番号を与える。
4	エラーの発生した$（ラベル）を与える。
5	カッコで囲んだCPUメモリのアドレスのデータの値を与える。
6	X座標だけを指定する。
7	カッコで囲んだVRAM（画面用メモリ）のアドレスのデータの値を与える。

	コメント
1	カッコで囲んだ数値の絶対値を与える。
2	メモリの使用サイズを、バイト単位で与える。 例）FRE（0）……BASICの作業領域のサイズ FRE（1）……残りのメモリ数
3	カッコで囲んだ数式で指定されるI/Oポート（入出力装置）からデータを1バイト読みこむ。
4	カッコで囲んだ数値を文字変数に与える。
5	0からカッコで囲んだ数までの乱数を発生する。
6	カッコで囲んだ数の符号を与える。 例）SGN（負の数）＝－1 SGN（ 0 ）＝0 SGN（正の数）＝1
7	本体の電源スイッチを入れてから現在の時間を秒単位で与える。65536秒周期

制御コード一覧表

キーボードの表示	10進数	16進数	は た ら き	シフトしたときの表示
	0	00	無 視	
A	1	01	無 視	[illegible]
B	2	02	カーソルのある行の先頭へカーソルを移動する	[illegible]
C	3	03	下スクロール	[illegible]
D	4	04	左スクロール	[illegible]
E	5	05	上スクロール	[illegible]
F	6	06	右スクロール	[illegible]
G	7	07	ベ ル	[illegible]
H	8	08	バックスペース	[illegible]
I	9	09	8ケタおきにカーソルを動かす	[illegible]
J	10	0A	1行送る	[illegible]
K	11	0B	カーソルをホームポジションへ戻す	[illegible]
L	12	0C	クリア・スクリーンをする	[illegible]
M	13	0D	RETURN キーと同じ	
N	14	0E	カーソルを今ある行の次の行の先頭へ移す	[illegible]
O	15	0F	標準モードへ移る	[illegible]
P	16	10	そう入モードへ移る	[illegible]
Q	17	11	マルチカラーモードへ移る	[illegible]
R	18	12	G(グラフィック)IIモードへ移る	[illegible]
S	19	13	G(グラフィック)Iモードへ移る	[illegible]
T	20	14	テキストモードへ戻る	[illegible]
U	21	15	表画面へ戻る	[illegible]
V	22	16	裏画面へ移り，裏画面での書きこみもできる	[illegible]
W	23	17	RETURN キーと同じ	[illegible]
X	24	18	カーソルのある行の右側を消す	[illegible]
Y	25	19	表示画面だけ切り換える書きこみはできない	[illegible]
Z	26	1A	書きこみ画面だけ切り換える。書きこんだ文字は表示されない。	[illegible]
[	27	1B	無 視	
¥	28	1C	右矢印	→
]	29	1D	左矢印	←
^	30	1E	上矢印	↑
−	31	1F	下矢印	↓

制御コードは、ダイレクトに使うときと、プログラム中で使うときとは、使い方が違います。

●ダイレクトで使うとき

CTRL キーといっしょに、制御コードの割りあてられている文字キーを押します。

●プログラム中で使うとき

PRINT命令と組み合せて使います。

例) PRINT " L " ←―― ダブルコーテーションマークで囲む

L ← CTRL SHIFT キーといっしょに押す。

また、制御コードは、CHR$でも使えます。

例) PRINT CHR$ (&C)

&C ← キャラクターコード表参照

ASCII（アスキー）コード表

CHR$で次の表の図形コードを指定する場合10進数と16進数で指定できます。16進数での指定では、&マークを付けます。

DEC →10進数
HEX →16進数

DEC	(HEX)	CODE	DEC	(HEX)	CODE	DEC	(HEX)	CODE
32	(20)		71	(47)	G	110	(6E)	n
33	(21)	!	72	(48)	H	111	(6F)	o
34	(22)	"	73	(49)	I	112	(70)	p
35	(23)	#	74	(4A)	J	113	(71)	q
36	(24)	$	75	(4B)	K	114	(72)	r
37	(25)	%	76	(4C)	L	115	(73)	s
38	(26)	&	77	(4D)	M	116	(74)	t
39	(27)	'	78	(4E)	N	117	(75)	u
40	(28)	(	79	(4F)	O	118	(76)	v
41	(29)	)	80	(50)	P	119	(77)	w
42	(2A)	*	81	(51)	Q	120	(78)	x
43	(2B)	+	82	(52)	R	121	(79)	y
44	(2C)	,	83	(53)	S	122	(7A)	z
45	(2D)	-	84	(54)	T	123	(7B)	{
46	(2E)	.	85	(55)	U	124	(7C)	¦
47	(2F)	/	86	(56)	V	125	(7D)	}
48	(30)	0	87	(57)	W	126	(7E)	~
49	(31)	1	88	(58)	X	127	(7F)	◀
50	(32)	2	89	(59)	Y	128	(80)	█
51	(33)	3	90	(5A)	Z	129	(81)	○
52	(34)	4	91	(5B)	[	130	(82)	♠
53	(35)	5	92	(5C)	¥	131	(83)	♣
54	(36)	6	93	(5D)	]	132	(84)	∼
55	(37)	7	94	(5E)	^	133	(85)	┻
56	(38)	8	95	(5F)	_	134	(86)	┳
57	(39)	9	96	(60)	`	135	(87)	┃
58	(3A)	:	97	(61)	a	136	(88)	━
59	(3B)	;	98	(62)	b	137	(89)	┫
60	(3C)	<	99	(63)	c	138	(8A)	┣
61	(3D)	=	100	(64)	d	139	(8B)	╋
62	(3E)	>	101	(65)	e	140	(8C)	┏
63	(3F)	?	102	(66)	f	141	(8D)	┗
64	(40)	@	103	(67)	g	142	(8E)	┓
65	(41)	A	104	(68)	h	143	(8F)	┛
66	(42)	B	105	(69)	i	144	(90)	▔
67	(43)	C	106	(6A)	j	145	(91)	▁
68	(44)	D	107	(6B)	k	146	(92)	▬
69	(45)	E	108	(6C)	l	147	(93)	■
70	(46)	F	109	(6D)	m	148	(94)	▍

DEC	(HEX)	CODE
149	(95)	▏
150	(96)	▎
151	(97)	█
152	(98)	◀
153	(99)	▼
154	(9A)	▲
155	(9B)	▶
156	(9C)	╭
157	(9D)	╰
158	(9E)	╮
159	(9F)	╯
160	(A0)	
161	(A1)	。
162	(A2)	「
163	(A3)	」
164	(A4)	、
165	(A5)	・
166	(A6)	を
167	(A7)	年
168	(A8)	月
169	(A9)	日
170	(AA)	時
171	(AB)	分
172	(AC)	ゃ
173	(AD)	ゅ
174	(AE)	ょ
175	(AF)	っ
176	(B0)	ー
177	(B1)	あ
178	(B2)	い
179	(B3)	う
180	(B4)	え
181	(B5)	お
182	(B6)	か
183	(B7)	き
184	(B8)	く
185	(B9)	け
186	(BA)	こ
187	(BB)	さ
188	(BC)	し
189	(BD)	す
190	(BE)	せ
191	(BF)	そ
192	(C0)	た
193	(C1)	ち
194	(C2)	つ
195	(C3)	て
196	(C4)	と
197	(C5)	な
198	(C6)	に
199	(C7)	ぬ
200	(C8)	ね
201	(C9)	の
202	(CA)	は
203	(CB)	ひ
204	(CC)	ふ
205	(CD)	へ
206	(CE)	ほ
207	(CF)	ま
208	(D0)	み
209	(D1)	む
210	(D2)	め
211	(D3)	も
212	(D4)	や
213	(D5)	ゆ
214	(D6)	よ
215	(D7)	ら
216	(D8)	り
217	(D9)	る
218	(DA)	れ
219	(DB)	ろ
220	(DC)	わ
221	(DD)	ん
222	(DE)	゛
223	(DF)	゜
224	(E0)	✽
225	(E1)	●
226	(E2)	♥
227	(E3)	◆
228	(E4)	∫
229	(E5)	╱
230	(E6)	╲
231	(E7)	╳
232	(E8)	＝
233	(E9)	╡
234	(EA)	╞
235	(EB)	╪
236	(EC)	◢
237	(ED)	◥
238	(EE)	◣
239	(EF)	◤
240	(F0)	▔
241	(F1)	▁
242	(F2)	▃
243	(F3)	▅
244	(F4)	▏
245	(F5)	▕
246	(F6)	▍
247	(F7)	▋
248	(F8)	▚
249	(F9)	▞
250	(FA)	⧗
251	(FB)	⋈
252	(FC)	◢
253	(FD)	◥
254	(FE)	◣
255	(FF)	◤

エラーメッセージ一覧表

エラーメッセージ一覧表

メッセージ	エラーの内容	エラーの原因	エラーの例又は対策
Err 1	FOR~TO~NEXT命令のエラー	●FOR命令に対してのNEXT命令がない。（FOR~TO~NEXT命令がペアーで使われていない。）	10 FOR A=1 TO 10 20 PRINT A 30 NEXT B 40 END 使われている変数を同じにする
Err 2	文法(ステートメント)エラー	●命令語を書くときに無い命令語を書いた。	PRIMT "C" スペルがちがう
Err 3	サブルーチンのエラー	●サブルーチン内でCLEAR命令を使った。 ●GOTO命令でサブルーチンへ飛んだ。 ●RETURN命令だけあって、GOSUB命令がない。	10 GOTO 30 20 END 30 PRINT "A" 40 RETURN サブルーチンへGOTO命令で飛んだ 行番号10をGOSUBにする。
Err 4	READ，DATA文のエラー	●DATA文のデータ不足。 ●READ文に対するDATA文がない。	10 READ A,B 20 DATA 44 データが1つ足りない
Err 5	引数の値の範囲がおかしい	●関数に与える値が、あつかえる数以外だとか、数を与えなければいけないのに、文字を与えてしまったとき。	10 A=RND(A$) 直接数値を与えるか、数値変数にして、与えなければいけない。
Err 6	オーバーフロー	●かけ算とるい乗の計算のとき、答えが極端に大きくなったとき。又は、極端に小さくなったとき(負の数)	PRINT 678*789
Err 7	メモリーオーバー	●変数を多く使い過ぎたとき。 ●サブルーチンを多く使い過ぎたとき。	DIM A(10000) 変数を多く取りすぎた。
Err 8	指定の行標がない	●GOTO命令やGOSUB命令での行き先がない。	10 GOTO 100 とした時に行番号100がない
Err 9	配列変数のエラー	●配列宣言時の間違い。 ●（　）の中の数が負か0。	DIM A(0) DIM B(-1) （　）の中の数値がおかしい
Err 10	配列変数のエラー	●同じ変数に2度配列宣言をした。	10 DIM A(6),A(3) 同じ変数を使っている
Err 11	0による割り算の実行	●わり算をするときに、0で割ってしまった。	PRINT 5/0 0では割れない

Err 12	ダイレクト実行命令が不適当	●ダイレクト実行（プログラムに組まない実行）命令で、間違ったことをさせた。 ●継続実行が不能なときにCONT命令をさせた。	PRINT 5/0 間違っている （文字変数としてあつかうならダブルコーテーションマークで囲む）
Err 13	データの型が合わない	●データの形式が違うものをいっしょに使った。	データの形式を同じにする
Err 14	スタックオーバーフロー	●文字列用のメモリが不足した。	文字列を少なくする
Err 15	文字列に関するエラー	●文字列が長すぎる ●計算途中で文字列の長さが19字以上になった。 ●代入において、左辺の変数よりも大きい文字列を代入した。	10 A$="abcdefghij" 20 B$="klmnopqrst" 30 C$=A$+B$ 18字を越えてしまう
Err 16	配列変数のエラー	●宣言されていない配列を使用した。	配列宣言を行う
Err 17	$（ラベル）の２重定義	●ジャンプ先に同じラベル名が２つ以上使われている	10 GOTO $A 50 $A 70 $A ２つ同じラベル名が使われている。どちらかにジャンプしていいかわからない。

BASIC I テキスト

●1982年11月　　　　　　初版発行

●発行所　株式会社ソード電算機システム

　　　　東京都葛飾区西新小岩4-42-12

●落丁・乱丁本はお取り替えいたします。

TAKARA

ご不明の点がございましたら、下記サービスセンターまでお問いあわせください。

☆本社サービスセンター　TEL.直通(03)602-3030

東京都葛飾区青戸4－19－16　〒125（㈱タカラ本社内）

TAKARA